(2)

MODELLING A COMBAT VEHICLE COMMANDER WITH AN EXPERT SYSTEM

by
Captain Timothy J. Gibson
B.S., United States Military Academy,
West Point, N.Y. 1979

Submitted to the Computer Science Advisory Committee and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Masters of Science with a major in Computer Science.

Dr. Jerzy Grzymala-Busse
Professor in Charge

Dr. Alkiviadis Akritas
Committee Member

Dr. Dov Dori
Committee Member

For the Graduate Committee

Date Thesis Accepted

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS: N/A
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: CPT Timothy J. Gibson / 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code): 124 James, Bethalto, IL 62010	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: EE Dept / 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): USMA, West Point, NY 10996	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification): Modelling a Combat Vehicle Commander with an Expert System (Unclass)

12. PERSONAL AUTHOR(S): CPT Timothy J. Gibson

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM Aug 88 TO Apr 89	89 Apr 15	199

16. SUPPLEMENTARY NOTATION: Prepared to Complete MS in Computer Science

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Artificial Intelligence; Robots; Robotics; (see reverse)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis examines and models the decision making process of a combat vehicle commander. The model uses the MYCIN certainty factor method. The thesis provides an overview of the following topics: expert systems; certainty factor theory and certainty factor computation; combat decision-making; uncertainty in battle; sensor capabilities for the system to "see"; the methodology of modelling a combat vehicle commander's thought process; and implementing the model with an expert system. The model is implemented with a commercial expert system shell called M.1.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: Unclass
22a. NAME OF RESPONSIBLE INDIVIDUAL: CPT Tim Gibson	22b TELEPHONE (Include Area Code): (618) 377-8565 / 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

A

ABSTRACT

This thesis examines and models the decision making process of a combat vehicle commander. The model uses the MYCIN certainty factor method. The thesis provides an overview of the following topics: expert systems; certainty factor theory and certainty factor computation; combat decision-making; uncertainty in battle; sensor capabilities for the system to "see"; the methodology of modelling a combat vehicle commander's thought process; and implementing the model with an expert system. The model is implemented with a commercial expert system shell called M.1.

TABLE OF CONTENTS

Chapter 1
The involvement of the American Military
in the development of Computers.....1

Chapter 2 An Overview of Expert Systems
and Certainty Theory.....5

Chapter 3 Why an Expert Gunnery System?.....22

Chapter 4 Current Bradley Gunnery Methods
and Systems.....28

Chapter 5 Uncertainty in Battle.....54

Chapter 6 Gathering Information - Sensors.....59

Chapter 7 Modelling a Gunnery System.....65

Chapter 8 The M.1 Programming Shell
and Inference Engine.....71

Chapter 9 The Expert Gunnery Model.....79

Chapter 10 Conclusions.....92

BIBLIOGRAPHY.....96

APPENDIX I Program Source Listing.....100

APPENDIX II Knowledge Base Listing.....122

APPENDIX III An Expert Gunner
Consultation Session
Without a Knowledge Base Trace.....149

APPENDIX IV An Expert Gunner
Consultation Session
With a Knowledge Base Trace.....166

APPENDIX V Inference Tree For
the Expert Gunner Knowledge Base.....187

Chapter 1 The involvement of the American Military in the development of computers.

The American Military has been involved in the development of computers from the early days of ENIAC during the Second World War. The military has consistently supported advanced research into new areas of computer science. The funding and direction of the military led to a number of new technologies and theories being introduced. The applications for these new technologies or theories have normally been at the strategic level. ENIAC was developed to solve ballistic equations for gunnery problems. The Colossus machines in England were developed solely for military codebreaking. Military codebreaking activities in the United States gave birth to entirely new computer technologies [Williams 85].

Military computer use continued through the 1960's and early 1970's with a computerized air defense system, the Semi-Automatic Ground Environment (SAGE) [Augarten 84], computerized ballistic missile systems, and computerized navigation systems. But all of these computer applications are far from the viewpoint of the ordinary soldier.

Military computers are used for essentially the same things that other people use computers for. Computers are very good at mathematical calculations, repetitive data manipulation, and storing and retrieving data. The military needs these tasks done just as much as any large corporation. Some tasks like string manipulation for codebreaking and math calculations for nuclear research would be impossible without computers. However, until recently, the cost associated with computers made them impractical for use at lower than strategic levels.

The cost of computers eventually began to decrease. In the late 1970's the U.S Army began fielding computers to assist artillery units in the computation of fire control data for artillery pieces. Although these computers are large and slow by today's standards, at the time they were revolutionary. These fire direction computers relieved men of the tiresome task of computing the proper elevation and deflection for the artillery pieces when firing at a distant target.

After the success of these fire control computers for the artillery, other branches of the Army joined the computer age. Tank fire control systems went from

analog "computers" with mechanical linkages to digital computers with sensors and laser rangefinders. Tactical air defense units were equipped with computerized warning systems.

Today the military is making use of computers to automate repetitive jobs at even lower levels. Broken or damaged equipment and repair parts are tracked with computers. Tactical units send status reports and orders through a radio linked computer network. "Smart" munitions are able to see and select their own targets. In essence, each one contains a small expert that decides which target is best to attack. These smart munitions decide on their own what to attack and in some cases where to fly to conduct the attack.

The biggest challenge, and the biggest potential payoff for the military in the future, is in providing more complex expert systems. These new expert systems will be able to advise or assist in more complex tasks and situations. Expert systems can assist commanders and staff officers in making decisions and formulating plans. In addition, they can assist soldiers in troubleshooting equipment failures. Expert systems can also replace men in some hazardous duties.

As an example, the Defense Advanced Research Projects Agency (DARPA) is currently sponsoring experiments with Autonomous Land Vehicles (ALV) that power and guide themselves from one point to another. The expert system in this case replaces the driver in the vehicle and makes the same decisions and reacts in the same way a human driver would. Autonomous vehicles could possibly replace men in either routine or hazardous driving duties. Implementing these ALV's is a very difficult job. A driver must see and analyze the terrain, and then decide where to go. These tasks are very simple for a human accustomed to driving, but for a machine the job is staggering.

Before going further in discussing applications for expert systems, it would be appropriate to take an in depth look at what an expert system is.

Chapter 2 An Overview of Expert Systems and Certainty Theory.

Expert Systems - The development of computers has followed many paths over the last forty years. One of these paths involves making computers act more like humans. These efforts at tracing and imitating human decision-making, making robotic devices mimic human counterparts, and synthesizing human speech are called Artificial Intelligence. [Harmon 85]

A recent outgrowth of Artificial Intelligence is the Knowledge-Based Expert System, or Expert System for short. An expert system provides human expertise in a specific area. Expert systems are particularly well suited to areas where there are few human experts or the cost of an expert outweighs the cost of an expert system. [Grzymala-Busse 88]

Regardless of how complex these expert systems are, they are still programs implemented on machines. They cannot be creative. If a problem calls for a new type of solution, the expert system may very well fail. However, within their area of expertise (or domain), expert systems can be relied upon to provide the user

with good, consistent advice. One of the more common expert systems is a rule based expert system.

The rule based expert system provides advice by applying rules to a set of facts. The rules are programmed in upon the advice of a subject matter expert, and a set of facts may come from the user or from a database. The outcome of these rules and facts is a conclusion. For example:

A rule. **IF** it is raining **THEN** take an umbrella.
A fact. it is raining.

This rule and this fact taken together would lead you to conclude that you need to take an umbrella.

You can also have multiple rules which may be related in their conclusions. These rules can be used in a **chain** to find more than one result. For example:

Rule 1. **IF** it is windy **THEN** it is raining.
Rule 2. **IF** it is raining **THEN** take an umbrella.
Fact. it is windy.

From the fact that "it is windy" and the two rules given, you can conclude from

Rule 1) it is raining
Rule 2) take an umbrella.

This process of taking one or more facts, applying them to rules, and then finishing with a **set** of results is called **forward chaining.**

Rules can also be **backward chained.** Backward chaining is accomplished by presenting the system with a goal (i.e., a desired result) and then seeing if that goal is achievable with the current rules and facts.

Rule 1.	**IF** it is windy **THEN** it is raining.
Rule 2.	**IF** it is raining **THEN** take an umbrella.
Fact.	you have an umbrella.
Goal.	is it windy?

Because you have an umbrella, then you can infer that rule 2 can be applied. Therefore, it must be raining. By a similar inference, you can conclude that it is windy. The goal in this example is reachable. Therefore, the rule based system will return a yes.

The important difference between forward and backward chaining is that forward chaining returns a **set** of results, which may be reached with the given rules and facts, while backward chaining returns a **yes** or **no** answer. A yes is returned if the goal is achievable with the given rules and facts, and a no is returned if it is not.

MYCIN Certainty Theory - One problem that an expert system may need to come to grips with is uncertainty. The facts and rules of a problem may not be distinct enough to yield either a simple set of results or a yes/no answer.

In Certainty Theory every fact has a measure of certainty associated with it. For a fact "A", this is denoted by C(A). The certainty of a fact reflects the belief of that fact being true. A fact's certainty can range from +1 (for absolute yes) to -1 (for absolute no). A value of 0 represents total uncertainty.

In addition to the certainties associated with facts, Certainty Theory also has a certainty factor (CF) associated with each rule. The value for this certainty factor is normally assigned by a subject matter expert, and has the same range of values
[-1,1] and meanings as a fact's certainty.

Certainty Theory was developed for the MYCIN system at Stanford University in the mid 1970's. The MYCIN system was designed to assist doctors in diagnosing a patient with a bacterial infection or meningitis. MYCIN allows rules or facts with different certainty factors to be

chained together into a final result[1]. [Buchanan and Shortliffe 84]

Calculating Certainty Factors - The simplest case of computing certainty factors is where there is a fact B with an initial certainty of 0 (i.e., C(B) = 0). A rule with the form

IF A **THEN** B **WITH** CF = X

where C(A) = 1 and X is the rule's certainty factor returns a new certainty for B (i.e., C(B)) equal to X. [Grzymala-Busse 88]

An expansion of the weather problem will illustrate this better.

Rule 1. **IF** it is windy **THEN** it is raining **WITH** CF = 0.7

Fact. it is windy with a certainty of 1.0.
Fact. it is raining with a certainty of 0.

Conclusion: it is raining with a certainty of 0.7 (70%).

The answer really being sought in this statement is "What is the certainty of B occurring if A is true?" or C(B/A). As we have seen, in this case where C(A) = 1

[1] Although the terms **"Certainty Factor"** or **"Certainty Theory"** can also apply to other expert systems or artificial intelligence methodologies, these terms will be used to indicate the MYCIN Certainty Theory in this paper unless otherwise indicated.

and C(B) = 0, the new C(B) equals the certainty factor associated with the rule.

However, this method is only good where C(B) is initially equal to zero. The certainty of B (the rule's conclusion) may already have a value associated with it. This value may be set by an expert or acquired from a previous rule[2].

When rules are chained together, the conclusion's prior value needs to be taken into account. Chaining rules together is controlled by the following equations.

$$C(B/A) = \begin{cases} C(B)+(1-C(B))*CF & \text{if both C(B) and CF are greater than or equal to 0.} \\ C(B)+(1+C(B))*CF & \text{if both C(B) and are CF are less than or equal to 0.} \\ \dfrac{C(B)+CF}{1-\min(|C(B)|,|CF|)} & \text{if C(B)\&CF are of opposite signs.} \end{cases}$$

where C(A) = 1
and
|X| equals the absolute value of X.
[Grzymala-Busse 88]

[2] This occurrence is common in a backward chaining system like MYCIN.

The end result is that the certainty C(B/A) always lies between +1 and -1. Generally speaking, if either C(B) or the CF equals +1, the rule's conclusion equals +1. Likewise, a -1 for C(B) or the CF yields a conclusion of -1. Notice that these formulas assume that the certainty of event A is equal to 1. [Buchanan and Duda 83]

If the certainty of A is between 0 and 1, the procedure in MYCIN is to multiply the certainty factor by the certainty of A.

Example:

Rule 1. **IF** it is windy **THEN** it is raining **WITH** CF = 0.7
Fact. it is windy C(0.6).
Conclusion. It is raining C(0.42).

There are times when the certainty of A may be less than 1. This may be caused by conclusion from another rule. If the certainty of A is less than 0, then the rule is not used. [Grzymala-Busse 88] and [Buchanan and Duda 83]

The MYCIN system makes an additional modification of this rule. In MYCIN, if the value of C(A) is less than 0.2, then the rule is not used. This decreases the use of rules which will have little impact and increases the systems speed. [Buchanan and Shortliffe 84]

An interesting aspect of certainty theory is that when evidence is chained together, the certainty of the conclusion at the end of the chain is always the same regardless of the order the rules are chained together. This is true as long as the same rules and facts are applied. The following examples will help to illustrate this point. [Harmon 85]

Rule 1.
IF it is windy **THEN** take an umbrella **WITH** CF = 0.6

Rule 2.
IF it is raining **THEN** take an umbrella **WITH** CF = 0.5

Fact 1. It is windy.
Fact 2. It is raining.

Given both of these facts and both of these rules, what is the certainty that you should take your umbrella? As discussed before, these two independent rules reinforce each other. Using the formulas discussed earlier, the following occurs.

```
Initial value C(B) = 0
Applying Rule 1 with CF 0.6 yields

C(B/A) = C(B)+(1-C(B)) * CF
       = 0 + (1-0) * 0.6
       = 0 + 1 * 0.6
       = 0.6
                          Result C(B) = 0.6

New value C(B) = 0.6
Applying Rule 2 with CF 0.5 yields

C(B/A) = C(B)+(1-C(B)) * CF
       = 0.6 + ((1-0.6) * 0.5)
       = 0.6 + (0.4 * 0.5)
       = 0.6 + 0.2
       = 0.8
                          Result C(B) = 0.8
```

Thus, by combining the two certainty factors for taking an umbrella, the final certainty is 0.8 or 80%. Applying the same rules in the opposite order to the equations produces the following results.

```
Initial value C(B) = 0
Applying Rule 1 with CF 0.5 yields

C(B/A) = C(B)+(1-C(B)) * CF
       = 0 + (1-0) * 0.5
       = 0 + 1 * 0.5
       = 0.5
                          Result C(B) = 0.5

New value C(B) = 0.5
Applying Rule 2 with CF 0.6 yields

C(B/A) = C(B)+(1-C(B)) * CF
       = 0.5 + ((1-0.5) * 0.6)
       = 0.5 + (0.5 * 0.6)
       = 0.5 + 0.3
       = 0.8
                          Result C(B) = 0.8
```

The final result, to take an umbrella, is still 0.8 even though the rules were combined in a different order. Thus, the final certainty is always the same regardless of the order the rules are applied as long as the same rules and facts are available. [Teknowledge 86a] This property allows models built with certainty theory to chain rules together as they are encountered and still return the same conclusion and certainty at the end of the chain. [Harmon 85]

Rules with Multiple Facts - So far all of the examples discussed have dealt with rules which have only one fact and one conclusion. However, many times an expert system must combine facts before making a conclusion, just as a person may consider more than one fact before deciding on a course of action. Computing the certainties for conclusions with multiple facts is done in a different manner than rules with only one fact. By way of example, we will modify the weather problem.

IF it is windy **AND** rain is forecast
THEN take an umbrella **WITH** CF = 0.9

In this rule, both the wind and the rain forecast need to be considered in deciding whether or not to take an umbrella. This "and" is a logical **AND** operator. Besides the logical **AND** operator, rules may also have a

logical **OR** operator, or a logical **NOT**. Calculating the certainty for multiple facts with these logical operators in it is based on the following formulas. [Teknowledge 86b], [Buchanan and Duda 83] and [Grzymala-Busse 88]

C(A **AND** B) = minimum of (C(A), C(B))

C(A **OR** B) = maximum of (C(A), C(B))

C(**NOT** A) = 1 - C(A)

where A and B are facts.

In the example, the certainty for taking an umbrella must be computed after the values for the combined certainties of A and B are known[3].

Example:

IF it is windy C(0.8) **AND** rain is forecast C(0.5)
THEN take an umbrella **WITH** CF = 0.9

where C(take an umbrella) = 0 initially

Because this rule uses an **AND** operator, the rule's certainty is computed using the minimum of the certainty of the two facts and the certainty factor of the rule. We select the lesser of the two certainties, in this case 0.5, and apply it to the formula

[3] The certainty of A and B may rest entirely on the outcome of another set of rules.

MIN (C(A), C(B)) = 0.5
CF' = (0.5 * 0.9) [4] = 0.45

C(C/(A **AND** B)) = C(C) + (1-C(C)) * CF'
= 0 + (1 - 0) * 0.45
= 0 + 1 * 0.45
= 0.45
Result C(C) = 0.45

Hence, the conclusion "take an umbrella" acquires a certainty factor of 0.45.

If this rule used a logical **OR** instead of the logical **AND**, the conclusion's certainty would be computed from the maximum of the certainties of the facts.

IF it is windy cf 0.8 **OR** rain is forecast cf 0.5
THEN take an umbrella WITH CF = 0.9

Selecting the value of 0.8 as the maximum, it is then applied to the formula

[4] Remember, if the certainty of A is between 0 and 1, the procedure is to multiply the certainty factor by the certainty of A.

MAX (C(A), C(B)) = 0.8
CF' = (0.8 * 0.9) = 0.72

C(C/(A **OR** B)) = C(C) + (1-C(C)) * CF'
= 0 + (1 - 0) * 0.72
= 0 + 1 * 0.72
= 0.72

Result C(C) = 0.72

Advantages and Disadvantages of MYCIN Certainty Factors

My presentation of the MYCIN Certainty Factor theory is not meant to be an unequivocal endorsement of it. As with any theory or method, the MYCIN Certainty Factor method has both advantages and disadvantages. In fact, the disadvantages of MYCIN certainty factors come close to outweighing their advantages. One of the best advantages of certainty factors is their simplicity. As seen in the previous sections of this chapter, calculating the values of certainty factors is relatively straightforward. The model is also attractive because it lends itself to estimates for certainty factors by experts. [Buchanan and Shortliffe 84]

Even more important, in my opinion, is the ease of being able to explain how these certainty factor values are computed. I do not believe that a user should blindly

trust the opinions or conclusions of an expert system. The user of an expert system should know the basic "line of reasoning" that the expert system is following. By this, I mean the user should know the general topics the expert system considers in its decision-making process and the general form the decision making follows.

MYCIN also has some distinct disadvantages. [Buchanan and Shortliffe 84]

1) MYCIN is not immune from any inconsistency introduced into the rule base by either the subject expert or programmer.

2) The MYCIN certainty factor computations do not relate to the rules of conventional probability theory. Additionally, a single piece of negative (or positive) information may overwhelm several pieces of positive (or negative) information.

3) MYCIN may discount important information by not using rules with facts of a certainty of less than 0.2. This is particularly true in cases where multiple facts are bound together with logical **AND** operators.

Example:

IF A **AND** B **AND** C **THEN** D **WITH** CF = 0.9

Depending on the implementation, if C(A) has a value of less than 0.2 the rule may not be used. The inference engine may stop when it sees three facts **AND**ed together with one less than the threshold value of 0.2. However, a backward chaining inference engine could effect the value of A while seeking B or C.

4) MYCIN certainty factors may not yield the correct inference if the inference comes at the end of long inference chain. [Adams 76]

5) Finally, MYCIN certainty factors are not the best environment to monitor a situation for a long period. The MYCIN model looks at a "snapshot" of the situation and makes a decision. If there is a need for long term monitoring, the model needs to be "restarted" at frequent intervals. [Buchanan and Shortliffe 84]

In addition to these problems, the MYCIN Certainty Factor method also has some distinct disadvantages when compared with other methods. The more common of these alternate methods are systems which use: Bayes Theorem,

Dempster-Shafer Theory, the INFERNO representation, and Fuzzy Set Theory.

The normal interpretation of Bayes Theorem provides a single probability of an event or rule conclusion given the probabilities of other preceding events. In this respect, Bayes Theorem is similar to MYCIN Certainty Factors. However, here the similarity ends. The formulas for calculating the probabilities for Bayes Theorem have a more theoretical and statistical foundation than those for certainty factors. The result is that Bayesian representations generally have more credibility than those using MYCIN Certainty Factors.

Other representations like the Dempster-Shafer Theory and the INFERNO representation work on a two value approach. This means that the conclusion of a rule normally does not have a precise value. Rather, the conclusion has an upper and lower bound associated with it. [Grzymala-Busse 88] The weather problem expressed with two values could take on the following form[5].

---

[5] This rule's form is for illustrative purposes only and does not conform with Dempster-Shafer theory, Inferno or Fuzzy Sets.

IF it is windy 0.7 OR rain is forecast cf 0.4
THEN take an umbrella [0.4, 0.7]

Thus, the conclusion "take an umbrella" does not have a precise value but is somewhere between 0.4 and 0.7. This is closer to the thoughts of a human expert because unlike the MYCIN Certainty Factors, rarely will a person say "I will take my umbrella 0.7 times if these facts are true".

All of these alternative representations have some advantage over MYCIN Certainty Factors. Bayes Theorem incorporates statistics better than Certainty Factors. Dempster-Shafer Theory, the INFERNO representation, and Fuzzy Set Theory all return two values for a conclusion and more closely approximate human reasoning.

Although the expert system in this thesis could have been done with any of these other systems I still chose to implement the model with MYCIN Certainty Factors. The reasons for this decision are discussed in detail in Chapter 9. For now, let me say that most of the reasons for this choice are of a practical nature rather than personal devotion to MYCIN Certainty Factors.

Chapter 3 Why an Expert Gunnery System?

With the introduction of devices like the Autonomous Land Vehicle discussed earlier, expert systems are moving out of a strict advising cycle and into an execution cycle. In this execution cycle expert systems are not only providing the advice but carrying it out as well.

Although expert systems are normally used when a human expert is too expensive, they can also be used when human reaction times or fatigue will have a detrimental effect on the performance of equipment. The U.S. Navy's Phalanx Defense System is a case in point.

The introduction of sea-skimming anti-shipping missiles (e.g., the Exocet or Harpoon) raised a new threat to surface ships. These missiles approach the ship at high speeds (300 - 500 mph) very close to the surface of the water. The result of this low approach at high speed is that the ship has only a few seconds from missile detection to missile impact. Human gunners are unable to stay alert for the extended periods required of an anti-missile defense system. Furthermore, they cannot see the incoming missiles during periods of low

visibility. The Navy introduced the Phalanx system to counter the threat posed by these sea-skimming missiles.

The Phalanx system combines a radar for gathering information and an automatic cannon for shooting down the incoming missiles. The Phalanx can be placed in an automatic mode in which it continuously scans approaching radar contacts, evaluates them to see if they should be considered hostile, and then shoots them down if they are. As such, the Phalanx is a good example of an expert system application. It has a limited domain (shooting down attacking missiles), and it provides a consistent response. It also provides something that men cannot give, alertness around the clock in any weather.

There are several good reasons to develop expert systems in land gunnery systems. First, land vehicle gunnery was one of the first areas to be automated by the military. Therefore, the military has experience with computers and electronics in tanks and other combat vehicles. Second, the gunnery domain is limited. This means that the tasks and responses required of a expert gunnery system are limited. These limited tasks and

responses are capable of being solved with a simple rule based expert system.

The third reason for developing an expert gunnery system involves the interaction of men, information, and machinery; a combat vehicle is a very tiring environment. It is common for soldiers to be awake for twenty hours a day for several days in a row. The soldier's senses, reflexes, and judgement degrade significantly in this environment. Anything which would assist a soldier in remaining more alert would be welcomed. The commander would not need to use such a system all of the time. However, when he is tired or otherwise occupied, an expert gunnery system would help him decide when to shoot at a target.

Gunnery is a term which can be applied to any heavy crew served weapon. Gunnery applies to all tasks the crew performs in firing their weapon(s). The term gunnery is particularly applicable in the areas of target detection, location, identification, weapon selection, and the decision to fire.

Whether the term gunnery is applied to a moving tank, a stationary armored personnel carrier, or an entrenched

missile launcher, the decision process is still essentially the same. The process of detecting a target, correctly identifying it, deciding what to do, and then carrying out the decision remains fairly constant regardless of the weapon.

Target Aquisition Process

Detection
Location
Recognition/Identification (Friend or Foe)
Weapon and Ammunition Selection
Fire Command

Execution of Fire Command

Table 1 [FM 23-1]

Because this decision making process is essentially the same for many different weapons systems, the end user is in many ways unimportant. However, the end user becomes important when specific facts must be applied to match specific weapon characteristics. This thesis is based on a working model of a gunnery system for a specific vehicle: the M2 Bradley Infantry Fighting Vehicle.

The primary purpose of a Bradley Infantry Fighting Vehicle (BIFV) is to carry a squad of infantry close to the tanks they support. Additionally, the BIFV is armed to destroy enemy tanks, other personnel carriers,

equipment, and personnel. Because many of these targets shoot at the Bradley, there is a high risk of the crew being killed or injured during battle. The men operating a Bradley are exposed to high levels of noise, fumes, and physical labor while operating it.

A good deal of knowledge is needed to operate a combat vehicle during a battle. However, most of it is learned and little of it is theoretical. The acts of driving and controlling a tank require the most knowledge. Detecting and firing at a target requires less knowledge. The Phalanx is an example of the latter. Loading the weapons requires almost no knowledge at all.

Table 2 shows three basic facts about machines, expert systems, and computers.

1) Machines are used to make work easier for men, or to take their place in repetitive or dangerous jobs.
2) Computers are very good at repetitive or time consuming calculations.
3) Expert systems are helpful in solving problems within a specific area of knowledge.

Table 2

These three facts combined with the characteristics of vehicle gunnery make it a good place to employ an expert system.

The vehicle gunnery system analysis in this thesis will concentrate on the tasks performed by the crew in the turret of the Bradley Infantry Fighting Vehicle. These tasks include all of the target detection, recognition, and engagement duties of the vehicle.

Chapter 4 Current Bradley Gunnery Methods and Systems.

Structurally, a Bradley Infantry Fighting Vehicle has a hull and a turret. The driver and a six man infantry squad ride in the hull. (see diagrams on pages 46 & 47)

The gunner and the vehicle commander ride in the turret. The turret also has most of the BIFV's weapons and ammunition. The nine men (seven in the hull and two in the turret) in the vehicle can be separated into two groups. The first group is a three man vehicle crew consisting of the commander (turret), gunner (turret), and driver (hull).

The second group is the vehicle's six man infantry squad. This infantry squad are passengers in the rear of the hull. They primarily perform tasks when dismounted from the vehicle. These dismounted tasks will not be addressed in this thesis. Only the vehicular tasks performed by the driver, gunner and commander will be discussed. Listed below is a summary of the tasks performed by the vehicular crew. [FM 17-12, FM 23-1]

Crew Duties

Commander

Communicate with higher headquarters

Detect and identify targets.
Decide whether or not to fire at enemy targets.
Issue initial fire commands.
Traverse turret to target if needed.
Command Gunner to fire.
Observe and adjust fire.
Issue a ceasefire command when the target has been lost or destroyed.
Alternatively request and adjust artillery fire.

Issue commands to driver

Gunner

Control automatic cannon, anti-tank guided missile system, and coaxial machinegun.
Determine range of target.
Fire and adjust weapons.
Operate independently of commander if required.

Driver

Provide maximum protection by protective terrain driving.
Maintain a stable platform for firing.
Follow Commander's instruction's.
Stop smoothly on command.

Infantry Fighting Vehicle Weapons

A Bradley Infantry Fighting Vehicle has a high velocity 25 millimeter automatic cannon as its primary weapon. The cannon can fire two types of combat ammunition: Armor Piercing Discarding Sabot (APDS) and High

Explosive Incendiary (HEI). The 25mm cannon is used against other lightly armored vehicles, and unarmored targets beyond the range of the machinegun.

The BIFV is also armed with a Tube-Launched, Optically Tracked, Wire Command-Link Guided (TOW) missile system. The TOW missile system provides the BIFV with the capability of destroying tanks and other armored vehicles from as far away as 3,750 meters (approximately 2.3 miles).

Additionally, the BIFV has a medium machine gun mounted parallel to the automatic cannon. The medium machinegun allows the gunner to engage personnel or unarmored targets without using automatic cannon ammunition. The medium machinegun is effective to 900 meters.[FM 23-1]

To fully understand the capabilities and limitations of the automatic cannon and the missile launcher, the reader should understand the different types of cannon ammunition and how the missile works.

Types of Automatic Cannon Ammunition

Armor Piercing Discarding Sabot (APDS)

This projectile is a kinetic energy weapon (see diagrams on pages 48 & 49); the damage to the target is derived completely from a high speed mass impacting on the target.

The projectile consists of a long, dense penetrator surrounded by lightweight metal petals. The petals allow the projectile to fill the entire bore of the cannon. This allows the hot gasses from the propellant to act on the entire diameter of the projectile when the cannon is fired.

Although the penetrator is very heavy, the entire projectile is relatively light. The projectile reaches a high velocity inside the cannon tube. Once the projectile leaves the cannon's bore, the metal petals fall away and the penetrator moves on its own at a high speed.[6]

The penetrator has a small diameter or cross-section of about 10mm, and a mass of 105 grams (approximately one-quarter pound). This large mass to diameter ratio gives

[6] Muzzle velocity for the APDS projectile is 1345 meters/second (4450 feet/second).

the penetrator a high cross sectional density. Because of the high velocity and high cross-sectional density the APDS projectile loses little velocity as it travels; there is little need for correction by the gunner for range or target speed. Several hits on a lightly armored enemy vehicle from APDS projectiles will normally destroy the enemy armored vehicle or kill the crew. The maximum effective range for this ammunition is 1,700 meters.[7] [FM 23-1]

High Explosive Incendiary (HEI)

This projectile is a standard explosive shell (see diagram on page 50). It consists of a fuse, body, and explosive filler. When the shell impacts, the fuse detonates the explosive filler, which in turn shatters the shell body. Damage is produced by concussion, blast, incendiary effects, and high speed metal fragments.

The HEI is used against unarmored vehicles, antiarmor guided missile positions and crew served weapons

[7] Maximum effective range is defined as the range at which a weapon has a 50% chance of striking the target on the first shot or burst.

positions. A HEI projectile is not effective against tanks. The HEI round has a maximum effective range of 3,000 meters. A timer will detonate the HEI projectile if it has not exploded before 3,000 meters. [FM 23-1]

TOW Guided Missile and Warhead

The missile consists of a fuse, a warhead, a flight control computer, eight guidance and flight fins, two wire spools, and rocket motors mounted in a missile body (see diagram on page 51).

The missile has a small infrared lamp in its base. The sight unit on the BIFV has an infrared tracker and an optical sight unit. The gunner tracks the target in his sight and fires the missile. When the missile is fired, the infrared tracker in the sight detects the infrared lamp in the missile base. A computer in the sight senses where the missile is (from the lamp) and where the sight is pointing and makes the angle between the missile and the sight go to zero by adjusting the missile's flight path. The corrections are passed to the missile through the two wires attached to the missile's base.

The gunner continuously tracks the target until the missile hits or misses the target. The TOW has a very high probability of hitting a target (90% at 3,000 meters), but it is also very slow (200 meters per second). The maximum range for the missile is 3,750 meters and the minimum arming range is 65 meters.

High Explosive Anti-Tank (HEAT) Ammunition

The TOW missile system only has one type of ammunition. The HEAT projectile is a chemical energy weapon. The damage to the target is derived from a high speed jet of molten metal being expelled at the target (see diagrams on pages 52 & 53)

The warhead is a cone shaped copper liner with an explosive charge behind it. When the warhead is fired, the explosive forces the copper cone into a long thin stream. This stream is about 10mm in diameter, moves at 3,000 - 10,000 meters/second, and will burn its way through most standard tank armor and any light vehicle armor. An advantage of the HEAT projectile is that the impact velocity has no effect on the warhead's performance. [MICOMb, FM 17-12]

A disadvantage of the HEAT projectile is the warhead's sensitivity to distance from the target. The warhead must be very close to the target (.5 meter) when it is exploded. If it is too far away the metal stream will scatter. If it is too close the stream will not be formed properly. An extendable rod on the nose of the missile ensures the proper distance before exploding the warhead. [FM 23-1]

Fire Direction Equipment

A BIFV is a complex and expensive piece of equipment. It is a fast, maneuverable, and comparatively well protected vehicle.

The BIFV is equipped with an optical rangefinder (soon to be upgraded with a laser rangefinder) and an Integrated Sight Unit (ISU). The ISU contains the optics needed to obtain the range to the target, and fire the automatic cannon, the missile system, and the machinegun. The ISU has magnifying optics and also has a passive infrared viewer to allow the gunner to detect and engage targets during darkness and periods of limited visibility. [FM 23-1]

The purpose of the automatic cannon is to get a projoctile to impact on a target. Automatic cannon fire is very accurate. A BIFV has a 50% chance of hitting another personnel carrier (target area 3.0 meters X 2.0 meters) with the first APDS burst at 1700 meters. Because of the ability to adjust from the strike of the first burst, the probability of the second burst hitting the target increases to over 90%. The HEI round does not have as flat a trajectory or move as fast as the APDS; however, the gunner can normally hit the target with the second burst of HEI fire.

The machinegun is an area weapon which is used to engage dismounted troops. The machinegun does not rely upon a one shot-one target prinicipal like the TOW missile. The gunner shoots at an area or group of dismounted troops when using the machinegun. He sweeps the gun over the target area to cover the entire area with bullets.

As mentioned previously, the TOW missile system is a very accurate weapon system capable of consistently getting first round hits at long ranges.

Crew Actions in Combat

This model is a simplified version of the Bradley Infantry Fighting Vehicle. The vehicular crew of a BIFV consists of a commander, gunner, and a driver. Although the duties of the driver are very important to a vehicle in combat, they will not be incorporated in this model. Implementing an expert gunner has been done before; for example, in the Phalanx system. This model will deal with a stationary gunnery platform and concentrate on the roles of the commander.

The vehicle commander's most important task is to find and evaluate targets for the gunner. The commander does this by constantly searching for movement, dust, and reflected light (from windshields, headlights, or armored vehicle periscopes). The commander focuses the search on likely areas of enemy activity, such as roads, trails, towns, and woodlines.

Once a target has been detected, the commander must recognize and identify the object. Listed in Table 3 is the model used by the U.S. Army Missile Command Electro-Optical Laboratory. [Felts]

Table 3 draws a distinction between the terms Detect, Recognize, and Identify. The differences between these terms are important and should be understood.

Activity	Observer Action
Detect	Detects that an object is within the field of vision.
Recognize	Recognizes the object and place it into a type. A type can be a tracked or wheeled vehicle, an artillery piece, or personnel
Identify	Properly identify the target by specific model. For example, a tracked vehicle would be properly identified if the commander knew it to be a British Challenger battle tank.

Table 3

Once the target has been properly identified the commander must decide what to do.

Engagement Options

There are many options for the vehicle commander to consider. First, the BIFV commander will try to report the enemy activity to his superiors. Then, the commander must decide if the enemy target should be fired at. Because firing at a target usually discloses the BIFV's location, the value of destroying the target must be weighed against the risk of exposure.

Alternatively, requesting artillery fire can often be a good option for the vehicle commander. The request is forwarded through artillery command channels to a firing unit. Artillery fire is relatively accurate and covers a large area. Artillery fire is very effective against soldiers on foot and unarmored vehicles. Artillery fire is less effective against armored vehicles even though special armor defeating munitions have been developed. After all, protection from artillery fire is why the British developed tanks in the First World War.

The following actions occur if the commander decides to shoot at the target with the automatic cannon.

(1) The commander alerts the gunner, tells the gunner to load the correct type of ammunition, and traverses the turret to the approximate direction of the target.

(2) The gunner then tries to locate the target through the gun sights. The gun sights have a more powerful magnification, but also have a more limited field of view.

(3) When the gunner is ready to fire he does so.

Engaging Tanks

Tanks are nearly always engaged with the TOW missile system because tanks are too heavily armored to allow the APDS projectiles to penetrate. The TOW missile is fired and tracked in the manner previously described. The TOW missile warhead is usually powerful enough to destroy most current armored vehicles with one hit.

Engaging Lightly Armored Vehicles

Lightly armored vehicles are normally fired at with the automatic cannon using APDS ammunition. When the cannon

is fired, the gunner follows the flight of the projectile to the target.[8]

If the projectiles miss the target, the next burst is adjusted towards the target. If the cannon projectile strikes the target, then the damage to the target is assessed. The gunner will continue to shoot at the target until he or the commander believes it is destroyed.

Lightly armored vehicles may also be engaged with the TOW missile system if they are beyond the maximum effective range of the automatic cannon.

Engaging Personnel

Personnel targets are not engaged in the same manner as vehicular targets. Normally, enemy soldiers are not as dangerous as enemy tanks and personnel carriers. Personnel are engaged with machinegun fire if they are within range of the machinegun. If the machinegun is used, the gunner follows the path of the bullets through

[8] The automatic cannon projectiles have a tracer element in the base to make this easier.

the use of tracers in the base of some of the bullets. The automatic cannon will be used if the enemy soldiers are out of machinegun range.

Caution needs to be exercised by the vehicle commander when engaging enemy personnel. Although enemy soldiers on foot are not usually a threat to the vehicle, they are very dangerous if they are close (less than 300 meters) or if they are armed with Anti-Tank Guided Missiles (ATGM's).

Engaging Unarmored Vehicles

Trucks and their relations are easy targets for combat vehicles. Unarmored vehicles have all of the weaknesses of dismounted personnel and none of the strengths. They are engaged in the same manner as dismounted personnel. If there are more truck targets than the machinegun can handle effectively, or if the targets are out of machinegun range, the vehicle commander may use the automatic cannon.

Anti-Tank Missile Guided Missiles (ATGM)

ATGM's pose a serious threat to BIFV's. ATGM's are very accurate and carry a powerful anti-tank warhead. The TOW missile is an ATGM. The disadvantage of ATGM's is that they fly slowly (200 - 400 meters/second) and must be tracked to the target by the gunner. Consequently, an ATGM's target can counter the ATGM by promptly firing at the launch location. If a vehicle is present then it is engaged appropriately. Otherwise, the commander must assume that the ATGM is being fired by dismounted infantry, and the launch area is fired at with the automatic cannon to suppress the infantry assumed to be there.

Other Targets

There are many known enemy targets which do not fit these categories. Supply complexes, command posts, and air defense sites are some of these. These targets can all be effectively engaged if they are considered as equivalent to a truck or infantry.

Engaging Unknown Targets

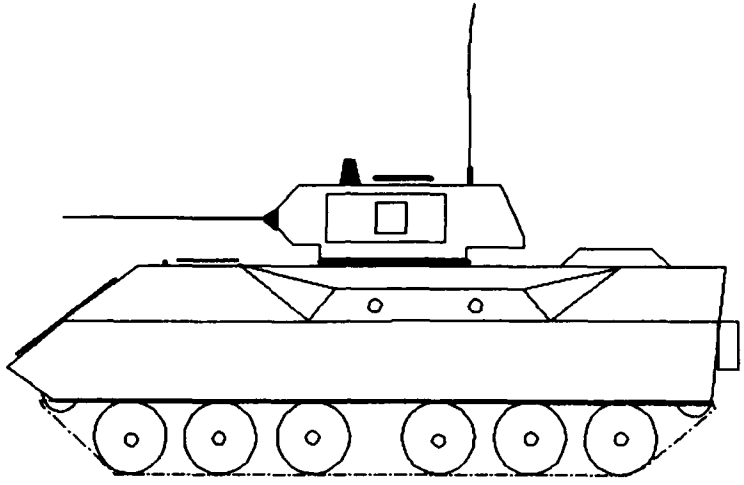
The last type of target is when a previously undetected target begins firing **at the vehicle the commander is in.** The commander observes the flash of a firing weapon and tries to identify the source. If the commander can identify the source, then it is engaged as previously described. If the commander cannot identify the source, then he must infer what the source is from its weapon signature. If this fails, the commander may infer that the target is probably whatever type of enemy unit is reported in the area. For example, if an enemy motorized infantry unit is in the area, the commander might conclude that the target is an armored personnel carrier.

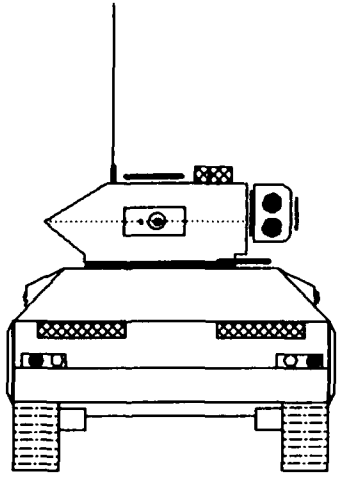
Once the decision to shoot is made, the engagement takes place as before. It is important **not** to fire at unknown targets unless they are firing at you. Friendly units may be in the area. When in doubt, find out.

Target Type	Range	Ammunition to Fire	Special Considerations
Any	>3.7km	None	Report and request artillery fire
Tank	65m-3.7 km	TOW	None
Tank	> 3.7 km	Artillery	Special Munitions
Armored Vehicle	0-1.7 km	APDS	None
	1.7-3.7 km	TOW	None
	> 3.7 km	Artillery	Special Munitions
Unarmored Vehicle	0-0.9 km	Machinegun	None
	0.9-3 km	HEI	None
	> 3 km	Artillery	Terrain dependent
Missile Launcher	0-3km	HEI	None
	> 3 km	Artillery	Terrain dependent
Personnel	0-0.9 km	Machinegun	None
	0.9-2 km	HEI	None
	> 3 km	Artillery	Terrain dependent
Other	0-3 km	As required	
Unknown	0-3.7 km	Any	Receiving fire Evaluate the area around the weapon firing. If no enemy are seen and the firing continues, fire at the weapon signature.

Table 4

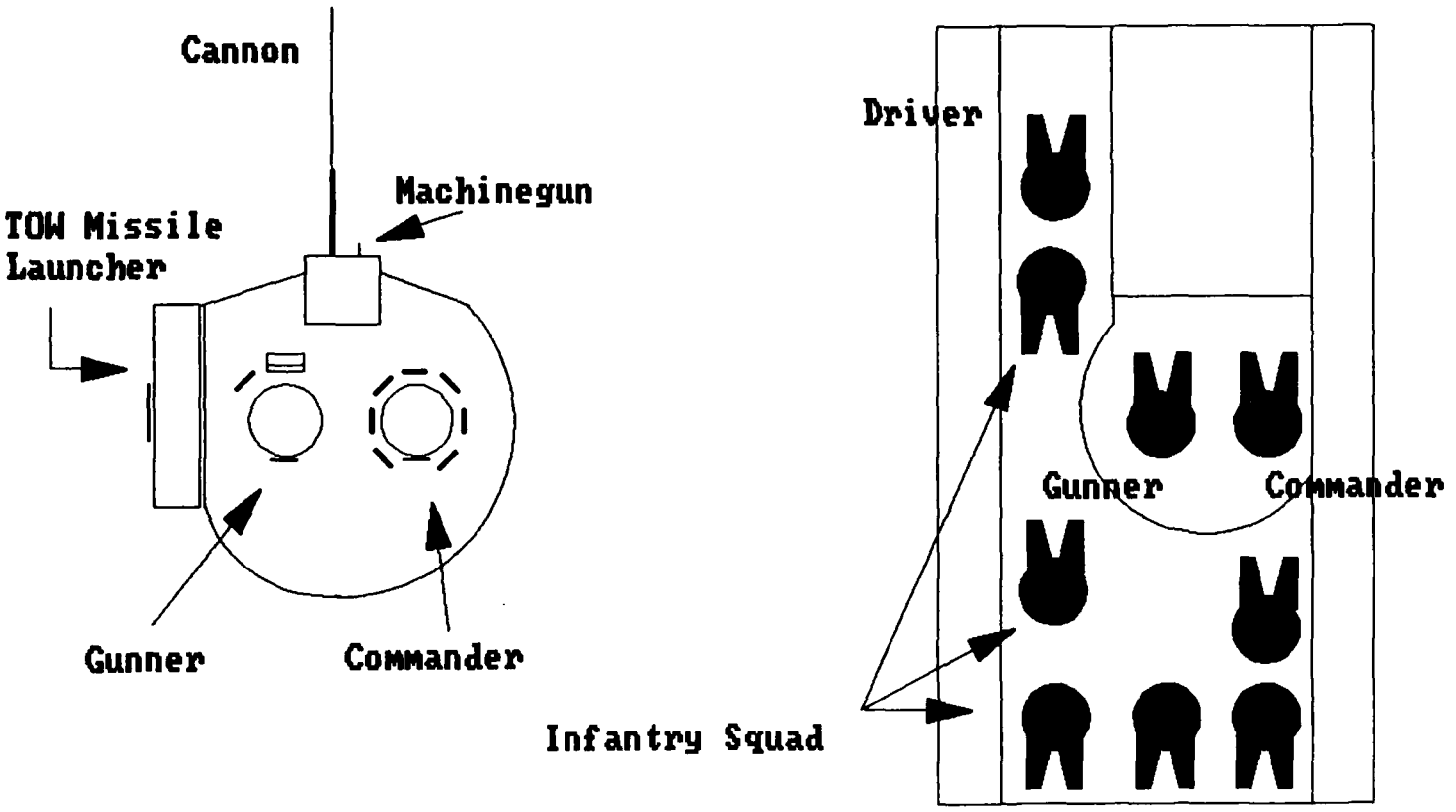
M2 Bradley Infantry Fighting Vehicle



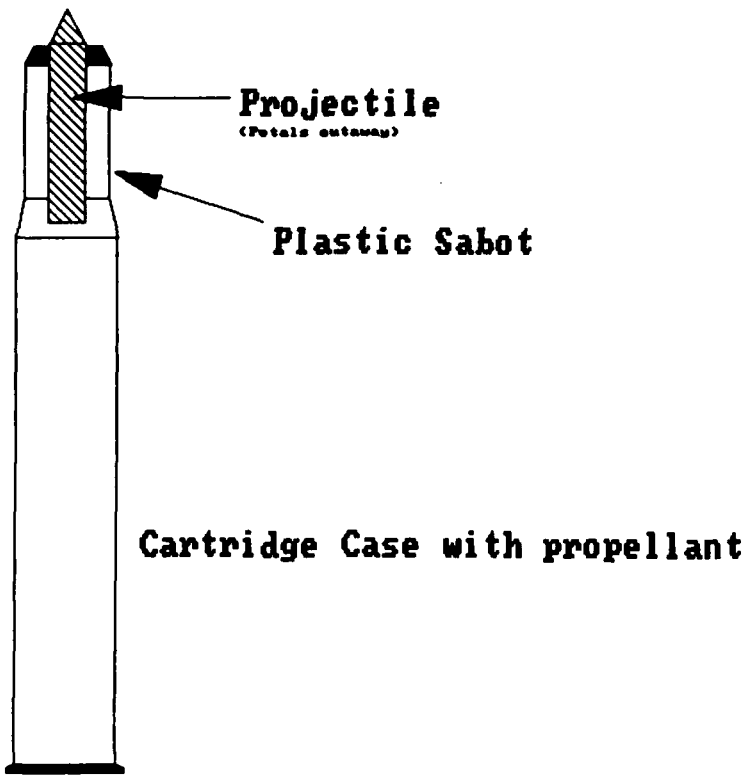


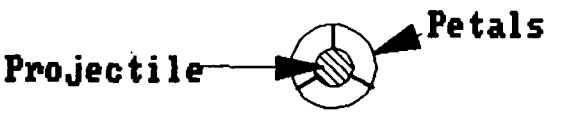
Weight	23.5 tons
Height	9' 8"
Width	10' 8"
Road Speed	40 mph
Cross Country Speed	30 mph
Cruising Range	300 miles

Bradley Turret and Personnel Seating



Armor Piercing Discarding Sabot (APDS)

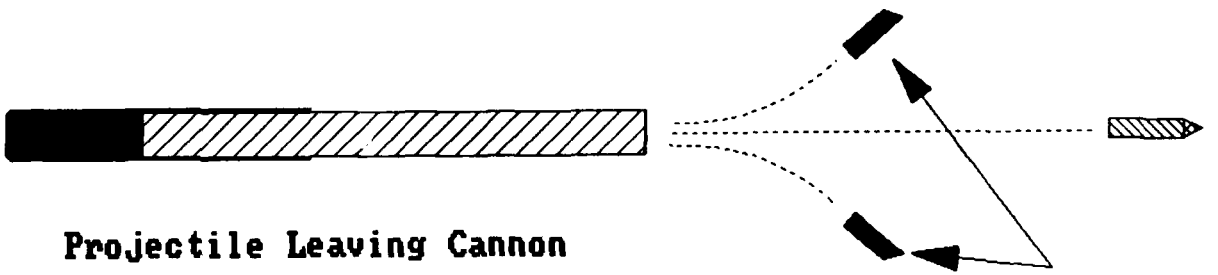




Front View of Projectile

Armor Piercing Discarding Sabot (APDS)

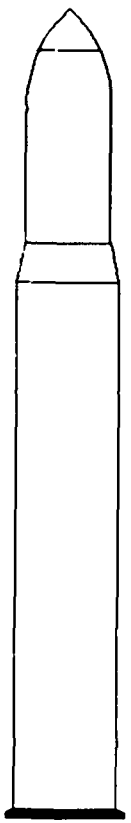
Projectile Fired and in Cannon



Projectile Leaving Cannon

Petals

High Explosive Incendiary Ammunition



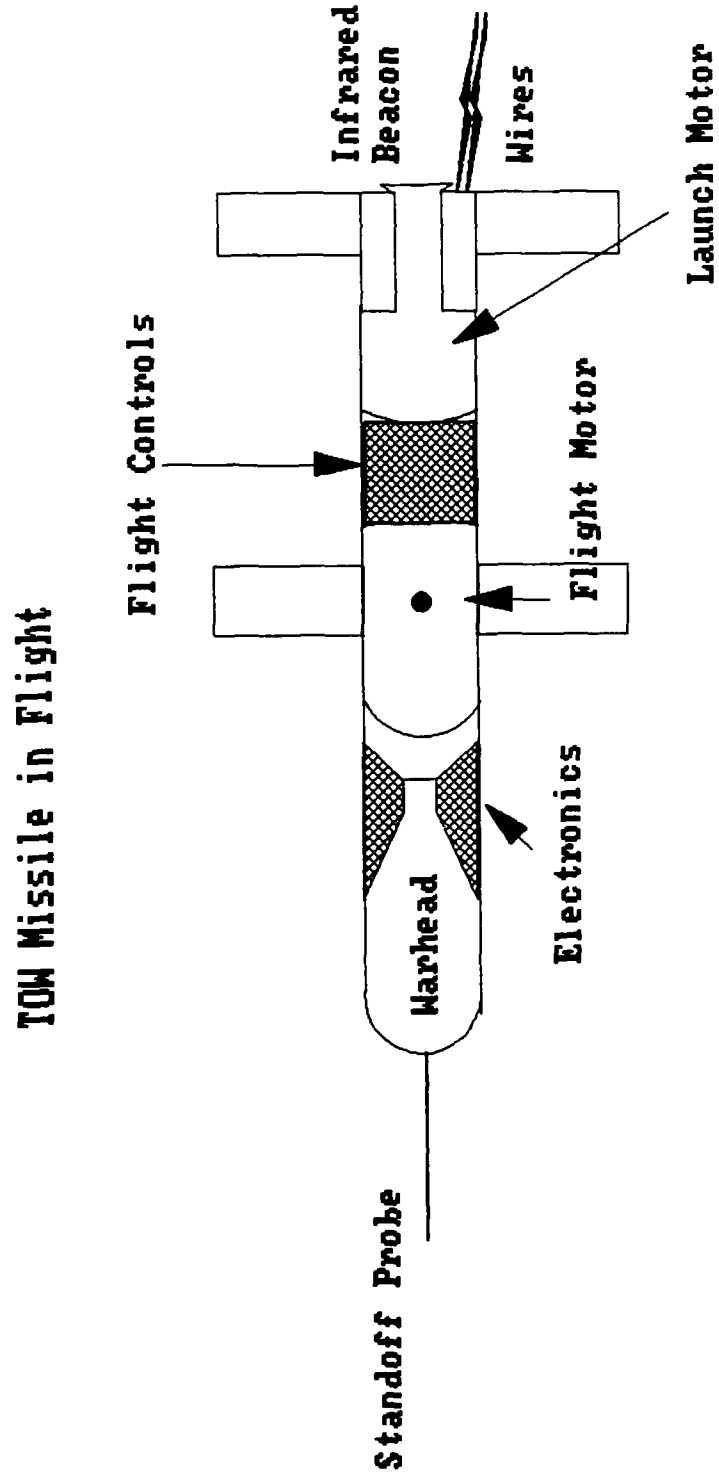
Projectile

Cartridge Case with propellant

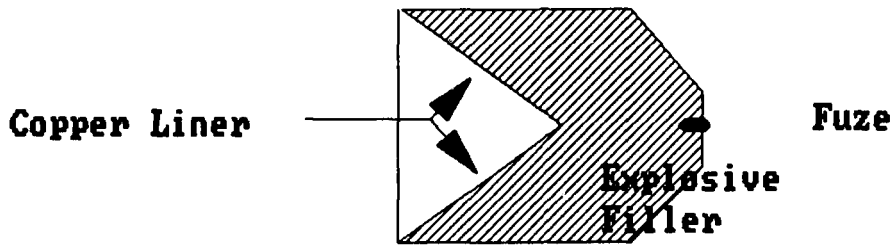
Fuze

Explosive Filler

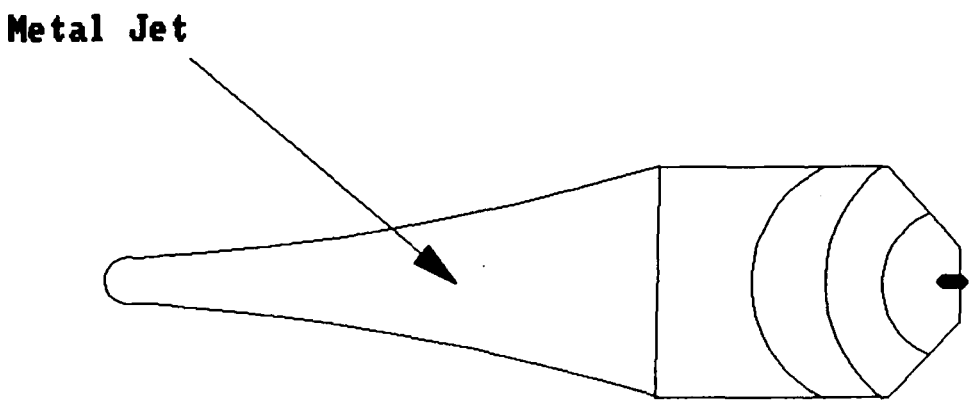
Shell Body


TOW Missile in Flight
Infrared Beacon
Wires
Launch Motor
Flight Controls
Flight Motor
Electronics
Warhead
Standoff Probe

High Explosive Anti-Tank Warhead

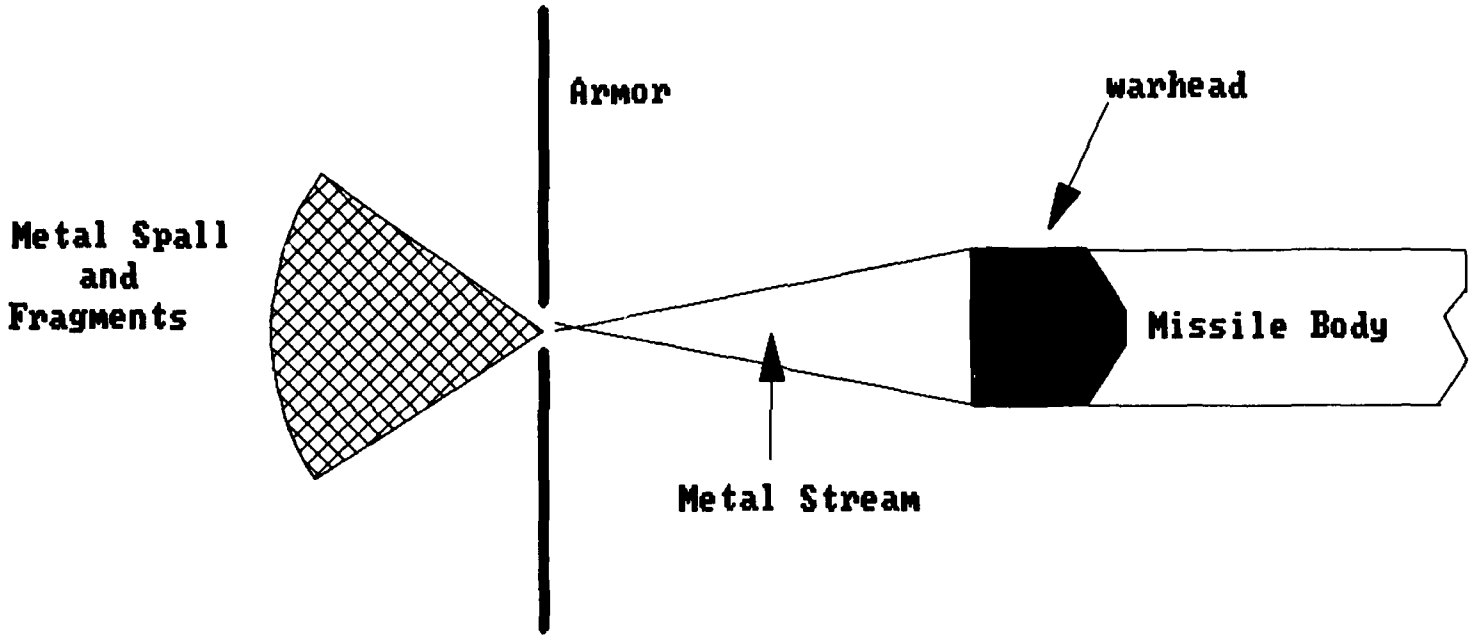


Before Firing



During Firing

High Explosive Anti-Tank Warhead



HEAT Warhead Impact

Chapter 5 Uncertainty in Battle

A Bradley Infantry Fighting Vehicle is a deadly, accurate piece of equipment. Hitting the target is not really a problem. The most difficult decision is deciding what to shoot at and what not to shoot at. For example, trucks are fair game, but ambulances are not; an unidentified tank may not be an enemy tank. Similar objects are easily confused.

Often, a combat vehicle commander must decide what to shoot at from limited information. Sometimes the commander will have a positively identified target, and he must only make the relatively easy decision of how to engage the target. It is in the areas of uncertain tactical information or incomplete target information that the commander has difficulty.

There are many factors the combat vehicle commander must consider before firing on an unidentified target. At a minimum the following items need to be considered.

The current tactical situation is important. If the friendly forces are attacking, there is a high probability that some friendly units are in front of the

BIFV. Then the question becomes what the target type is. If the commander knows that there are friendly **combat** forces ahead of him and he sees a truck, then he will probably fire at it. Similarly, he has a higher expectation of firing at a tank which is facing him than at a tank which is moving away. This is because a friendly tank to his front that is attacking should be following the latter course. An enemy tank should be facing him from a defensive position with the front of the tank facing him.

If the commander is the leading unit in an attack the decisions become much easier. The commander is now free to shoot at anything ahead of him, excluding noncombatants.

If the friendly forces are defending, a different set of rules needs to be applied. The first question is whether or not any friendly units are located in front of the commander. There may be security or reconnaissance units forward of the BIFV. If these are being driven back by the enemy, they will be approaching with the front of their vehicles facing the BIFV, just like the following enemy forces.

Knowing the current locations of enemy and friendly units is also important. A personnel carrier of unknown identity within a friendly defensive perimeter is probably friendly. On the other hand, any vehicle or movement within a known enemy position is probably an enemy and should be engaged.

There are certain places different types of vehicles are found. For example, trucks are very rarely found in or forward of the front lines. If a tank participating in an attack has travelled several kilometers past the enemy front lines, then the following would all be legitimate targets.

Unidentified trucks.
Unidentified artillery pieces.
Tents or command posts.
Signal installations
Air Defense units.
Missile units

These are all legitimate targets because of where they are. However, this legitimacy is based on **current** information. Information more than a few hours old can lead to incorrect assumptions.

The final factor to consider is target formation and terrain. Unknown vehicles moving cross country are probably combatants. The same applies to people on foot

moving cross country. Civilians will almost never move across country; they normally move on roads instead. Hence, it is a safe assumption that vehicles seen moving through a field in a combat zone are combatants. This assumption is reinforced if the people or vehicles are moving in a column or line. Additional reinforcement comes from the distance between individual people or vehicles. Soldiers normally travel in dispersed formations to reduce the damage caused by enemy fire. Civilians travel in close groups.

Objects on roads are not as easy to identify. The military uses roads almost as frequently as civilians, and looking for formations does little good. Traffic on a road is in a column regardless of whether or not it is military. The commander needs to have positive identification or apply one of the other factors previously discussed to fire at targets on a road.

Unfortunately, the application of the terrain and formation rules only tell the commander whether or not the targets are combatants. The use of terrain and formation rules will rarely tell the commander whose side the combatants are on.

The most difficult part of using these decision factors is combining them. Unidentified infantry following an identified enemy tank can be assumed to be enemy infantry. On the other hand, unidentified infantry shooting at an identified enemy tank can be assumed to be friendly.

Chapter 6 Gathering Information - Sensors

A combat vehicle commander collects information with his eyes and ears. The most important is his vision. He sees what is happening around him, and he listens to his radio for information concerning enemy and friendly activity.

The information collected by the commander supplements his existing information. An expert system assisting or replacing the tank commander and gunner would need similar information to make useful decisions.

Types of Electro - Optical Sensors

There are several different kinds of electro-optical sensors. The most common sensors use the infrared spectrum, both passive and active; active millimeter wave radar; and passive visible spectrum (television type) electro-optical systems.

A passive system amplifies existing energy, whether it is visible or not. Looking through a pair of binoculars is an example of passive information collection.

An active system is one that sends out some type of energy and then receives and interprets the return. Driving a car at night is an example of this. The light from headlights is the energy being sent out and your eyes and brain are the processors for the information being returned. Active systems are more susceptible to detection because they emit some type of energy. Detection may lead to enemy jamming or enemy attacks.

All of the sensors mentioned previously have strengths and weaknesses. By carefully choosing the sensors in groups that complement each other, a sensor "suite" can be made which will work in most conditions. Current "smart" systems use these sensor suites to provide good information and withstand countermeasures. One of the better combinations is Imaging Infrared and Millimeter Wave Radar. Both of these will be discussed in detail. [MICOMb]

Imaging Infrared (IIR) is a passive system which operates in a range with a wavelength from 1 millimeter to 100 micro-meters (1 - 300 terahertz). To give the reader an idea of where this is, the following table is provided.

Electromagnetic Emission	Frequency	Wavelength
Electrical Power	3-300 Hz	100 - 1 Mmeters
Radio Waves	10KHz-300MHz	10 Kmeters-1 meter
Microwaves	1-300 GHz	1 meter-1 mmeters
Infrared	1-300 THz	1 mm-1 micrometers
Visible Light		**.7-.4 micrometers**
		(red=.7 micrometers)
		(violet=.4 micrometers)
Ultraviolet		30-1 nmeters
X-rays		1 nmeter-10 pmeters

Hz = Hertz = Cycles per second
T = Tera = 10^{12}
G = Giga = 10^{9}
M = Mega = 10^{6}
K = Kilo = 10^{3}
m = milli = 10^{-3}
micro = 10^{-6}
n = nano = 10^{-9}
p = pico = 10^{-12}
[BDM]

Table 5

As you can see, the infrared spectrum is just below the red end of the visible light spectrum. Normal battlefield emissions in the infrared range include:

(1) fires and weapon discharges;
(2) vehicle engines, transmissions, and wheels/tracks; and
(3) body heat.

A tank viewed through an IIR thermal viewer appears as a light colored tank on a dark background. The engine compartment, the tracks, and the wheels appear to be brighter than the rest of the tank. The brighter appearance is caused by the higher temperature of those parts of the tank. The black background is from the cooler atmospheric temperature.

A thermal viewer works by having an IIR sensor cooled to cryogenic levels (77 degrees Kelvin) with a cooling system. The sensor then picks up the minute differences in temperature change, processes the signal and produces an image for viewing. [Felts]

IIR sensor systems work well during periods of darkness and are not effected by rain or smoke. Dust does degrade the performance of IIR sensors. A typical military IIR sensor allows vehicular target detection and recognition in excess of 3,000 meters. [MICOMb]

Millimeter Wave (MMW) radar operates in the lower end of the microwave frequencies near the infrared band. The MMW sensor consists of a transmitter, a receiver, and a processing unit. The transmitter sends out a signal which is reflected off of the objects it strikes. The

receiver picks up the returning signals and send them to the processing unit. The processing unit then produces an image from the returned signals.

By sending the returned signal to a different type of processor, the radar can compare the returned signals with "snapshots" of normal vehicle radar returns. Checking this table allows the radar to determine the target's identity. An active MMW in good conditions can correctly identify a target well past the ranges needed for this thesis. [Emmons 88]

Active MMW has the ability to accurately determine the range to targets and provides a better identification capability than any other sensor system. In addition, the narrow bandwidth of the MMW transmitter makes MMW less susceptible to jamming and more accurate at angular tracking than other systems.

The disadvantages of MMW radar lie principally in the fact that its transmission is severely attenuated by rain or heavy fog. For this reason a MMW sensor is often paired with an IIR sensor to provide an all-weather capability. [MICOMb]

If a fully automated gunnery system is ever implemented these sensors or something similar to them will be used. Sensors like these can also be expected on a gunnery system which detects and alerts a human vehicle commander.

Chapter 7 Modelling a Gunnery System

The most difficult aspect of implementing an automatic gunnery system is replacing the commander. Loading the correct ammunition and firing it at the correct target are comparatively easy. There are several systems currently available (e.g., the U.S. Navy's Phalanx Anti-Missile Defense System) which combine guns with radar to automatically engage and destroy hostile targets. However, all of these systems are anti-aircraft weapons. They will automatically engage **any** aircraft within range if they are allowed to.

To return to the task of the commander. If the sensor information available to the commander is clear and without uncertainty then the commander's job is relatively straightforward. However, land combat information is rarely clear or certain. Smoke, dust, vegetation, and enemy jamming combine to make information uncertain. It is this uncertainty that makes the commander's job difficult.

Many times, a commander must take whatever information he has and act on it. Even if the information is uncertain, the commander must make a decision. In this

gunnery model, the commander must decide whether or not to engage a target and what weapon and ammunition to use. An incorrect decision could either cost the lives of friendly troops or allow the enemy to win.

In the instances discussed before there are several common variables. The commander (or expert system) needs to consider these variables when he is making a decision on based incomplete information. The attributes are shown in Table 6.

General Information
Weapon and Ammunition Availability
Current Friendly Tactical Situation
Current Enemy Tactical Situation
Type of Enemy Unit known to be in the area
Status of Friendly Patrols
Status of Civilian Refugees
Specific Information
Target Side (Friendly, Enemy, Unknown)
Target Type
Target Range
Information for Uncertain Targets
Current Known Enemy Locations and Types
Current Known Friendly Locations and Types
Multiple Targets
Target(s) in Formation
On, near, or away from a road

Table 6

By considering this information the commander will decide whether or not to shoot. If the target is

clearly identified as an enemy, the commander will shoot. If it is friendly, the commander will not shoot. The problem lies between these two extremes.

If the target's side (friendly or enemy) or type is uncertain, the commander must ask himself questions about the target's attributes shown in the Table 6 for **Uncertain Targets.**

Current Systems

A problem with current systems is that they do not ask these questions of themselves. The Defense Department and the Army are both working on expert gunnery systems. The Defense Department work is being done by DARPA while the Army work is being conducted by the Army's Missile Command (MICOM).

Much of DARPA's work centers around case-based reasoning. Case-based reasoning is based on the following scenario.

A problem is presented to a system. This system searches its memory for a similar problem. It

"remembers" a previous problem that is similar to the current one. The system then focuses on the similarities between the new problem and the old problem. The solutions for these similarities are extracted from the old problem and applied to the new problem. [Kolodner 88a] If the new problem is not solved, the system applies rules to the remaining parts of the problem to completely solve it. [Koton 88]

This is the same reasoning process that many men use when they are presented with a everyday problems. The process has just been applied to a machine. The difficulty with the case-based reasoning process is that it consumes memory and micro-processors at a prodigious rate. In fact, the few working applications require par?llel processors or supercomputers to work adequately. [Kolodner 88b]

MICOM's gunnery systems are based on the millimeter wave radar systems discussed earlier. If a target is correctly identified then the system acts on that information. If a target cannot be identified then the system does not know what to do and it stops. The MICOM system does not incorporate uncertainty at all. The

target must be correctly **identified**[9] for the system to work. [Emmons 88]

However, it is not unreasonable for a system to ask for more information when it needs it. Sensors and databases are able to return answers to the questions a commander normally answers himself with his eyes and a map. A computer program should be able to control the sensors and databases to provide itself with the same information.

The model in this thesis is a rule based expert system that interacts with its sensor system to arrive at conclusions. The model does not ask any questions which are not available to either a vehicle commander or a computer with sensors and a database . Interactive expert systems are not new. The originality of this model comes from formalizing the gunnery process and developing a rule base for it.

The bulk of the work for this thesis is a program which implements Table 6. The program correctly selects the

[9] To identify means to correctly determine the target's type and model. For example, a correctly identified tank is a Leopard 2 of West Germany.

weapon and ammunition to fire at a known enemy target. It will not fire at a known friendly target. The program asks the user questions about the same topics covered in Table 6. The program uses the answers to these questions to determine whether or not it should fire at a target and if so, what weapon and ammunition combination is best. The program was implemented using the M.1 programming shell.

Chapter 8 The M.1 Programming Shell and Inference Engine

This section is included because although M.1 is designed to be very readable, M.1's control loops and the assignment of values to expressions is unlike any other language I have seen. This section will briefly describe the M.1 code. A full listing of the program is included at Appendix I.

M.1 is a product of Teknowledge Incorporated. It is written in the "C" language and was introduced to allow prototyping knowledge systems. The M.1 system can accommodate up to 2,000 rules and facts, and uses a backward chaining inference engine to arrive at its conclusions.

M.1 uses the same certainty theory as the MYCIN system discussed in Chapter 2. Rules and facts are represented in attribute-value pairs (called expressions) with associated certainty factors (cf). The only difference between the certainty factors of MYCIN and M.1 is that while MYCIN's range is from -1.0 to +1.0, M.1's range is from -100 to +100. [Harmon 85]

Control of a program in the M.1 shell is different from most other languages. M.1 is a backward chaining inference engine; therefore, it is trying to support a goal. A goal is specified at the beginning of the program, and all of the actions of the program are in search of that goal. Once the goal is found the program halts.

The user can also specify an item called "initial data". The initial data is a set of sub goals, and M.1 searches for all of the initial data conclusions before looking for the goal. An iterative program is accomplished by having only an initial data set and no goal.

An example from my program is shown below:

```
initialdata =
[begin the-consultation, continue the-consultation, all-targets-engaged].
```

With this initial data set, M.1 will solve for the three items

```
begin the-consultation
continue the-consultation
all-targets-engaged
```

in that order.

The goal begin the-consultation is a message telling the user what the program does, asking if he is prepared to continue, and halting if he is not. This is done with the only control statement available in the M.1 shell,

the IF...THEN statement. The IF section of the IF...THEN statements is able to do more than one operation or evaluation by having multiple expressions linked by Boolean "and" operators or Boolean "or" operators.

The THEN section can only have assignments to one expression.

Example:

```
if begin message = M and
      display(M) and
      begin signal and
      friendly-status is sought and
      enemy-status is sought and
      enemy-unit-type is sought and
      patrol-status is sought and
      civilian-status is sought
then begin the-consultation.
```

This control statement displays the message (see the first two pages of Appendix I), asks if the user is ready to continue (the expression begin signal), and then asks the user for information on friendly and enemy status, enemy units types, and the status of friendly patrols and civilian refugees. Execution of this statement stops if the response to begin signal is no. The next statement in the program is

```
if (not begin signal) and
    display("\t Tell the computer 'go' when you are ready to proceed \n") and
    display("\t or 'exit' to leave the M.1 shell.") and
    do(abort)
    then begin the-consultation.
```

The "do(abort)" statement aborts the program.

The "<expression> is sought" statement lets the program continue regardless of the value returned. For example, the expression

```
enemy-status is sought
```

lets the M.1 shell get a value for the expression enemy-status and continue, regardless of the returned value. This is useful when you only need a value assigned to an expression or when the answer is not a simple yes or no. The value for the expression enemy-status is sought by M.1 before it continues to the next statement. To do this, M.1 looks for enemy-status in the conclusion of other IF...THEN statements.

However, M.1 will not find the expression enemy-status in the conclusion of an IF...THEN statement. M.1 will then look to see if enemy-status is listed as a question. It is. The section of code below

```
question(enemy-status) =
["What is enemy's current tactical posture?"].
legalvals(enemy-status) = [attacking,defending,withdrawing,delay].
automaticmenu(enemy-status).
enumeratedanswers(enemy-status).
```

will generate the following information on the screen

What is enemy's current tactical posture?

1. **attacking**
2. **defending**
3. **withdrawing**
4. **delay**

M.1 allows the user a number of ways to respond. If the user wanted to respond with defending, he could answer with the number "**2**", the letter "**d**", or type out "**defending**". All have the same effect.

Certainty Factors in M.1

Certainty factors can be assigned explicitly with statements like

```
if self-defense  = yes
then should-fire = yes cf 100.
```

In this case should-fire is explicitly set to yes with a certainty of 100. Any previous value for should-fire is lost.

Certainty factors can also be modified in conjunction with responses or values such as

```
if the-side is sought and
        the-side = not_known and
        status-check = SC and
        do(set maybe-should-fire = yes cf SC) and
             .
             .
             .
        maybe-should-fire = Final
then should-fire = Final.
```

In this case the value of maybe-should-fire is updated using the MYCIN certainty method discussed in Chapter 2. The M.1 operator "set" is what allows the values for expressions to be adjusted using the MYCIN certainty method.

As you can see from the previous block of code, the expression maybe-should-fire is set to yes with a certainty factor of "SC". SC is a variable bound to the expression status-check and could be any value from -100 to +100.

Tracing through the operations on status-check will help explain both variables and facts in M.1.

In searching for a conclusion leading to status-check, M.1 will find this rule.

```
if friendly-status = FS and
        enemy-status = ES and
        status-check-function(FS, ES) = SCF
then status-check = SCF.
```

Both friendly-status and enemy-status have a value associated with them from previous questions. These values are passed to the fact status-check-function. There is a fact status-check-function for every possible combination or tuple of friendly-status and enemy-status.

Some of the 28 tuples are shown below:

```
status-check-function(leading, attacking) = 60.
status-check-function(attacking, attacking) = 55.
status-check-function(defending, attacking) = 60.
status-check-function(in_reserve, attacking) = 50.
status-check-function(withdrawing, attacking) = 50.
status-check-function(delay, attacking) = 50.
```

The number at the end of each tuple is the value which will be returned by status-check-function, and then returned to status-check. The expression status-check is then returned to the original IF...THEN statement where the expression maybe-should-fire is updated with the value returned for status-check.

The last function to be discussed is the reset function. This function is vital if the expert system written in M.1 is iterative in nature. If the entire program is always to be rerun, a simple command of do(reset) is

adequate. However, if background information is asked only at the beginning of a session, the reset command must be used selectively

Example:

```
if continue the-consultation and
        do(reset continue the-consultation) and
        do(reset self-defense) and
        do(reset should-fire ) and
             .
             .
             .
        do(restart)
then all-targets-engaged.
```

This fragment of code will reset continue the-consultation, self-defense and should-fire to be totally without value. Consequently, on the next iteration, their values will again be sought.

Finally, the command do(restart) will restart the consultation from the beginning. The consultation will seek values for all expressions that have been reset. The command do(restart) is just before the conclusion of the rule then all-targets-engaged. If you will recall from the beginning of this chapter, the program's initial data ended with all-targets-engaged. M.1 seeks this value to end the session, but if the user chooses to continue, the conclusion of the rule is never reached.

Chapter 9 The Expert Gunnery Model

An inference tree of this expert system is provided for the reader at Appendix V to help explain the workings of this expert system.

The initial data M.1 is seeking in this model is the expression begin the-consultation. As mentioned previously, this expression confirms that the user is prepared to continue and gets some preliminary data from the user. This includes the following information (shown with choices):

Current Friendly Tactical Situation: Leading (i.e., leading the attack), Attacking, Defending, Reserve, Reconnaissance, Withdrawing and Delaying.

Current Enemy Tactical Situation: Attacking, Defending, Withdrawing and Delaying.

Type of Enemy Unit known to be in the area: Tank, Motorized Infantry (Armor Personnel Carriers), Infantry.

Friendly Patrols in this area: Yes, No.

Civilian Refugees in this area: Yes, No.

Once this background information is gathered, the expert system asks if the target is shooting at the user. If the answer is yes, the expert system goes into a targeting phase where the target range and type are gathered. The range is input in meters and the target type is input as an enumerated question with these choices.

> Tank, Armored Personnel Carrier, Self Propelled Artillery, Towed Artillery, Anti-Tank Units, Infantry, Trucks, Logistics Areas, Command Post, Tracked, Wheeled, Dismounted, or Uncertain.

The input of a target type and a range lets the expert gunnery system select the best weapon and ammunition to destroy the target. Weapon and ammunition selections are made in accordance with Table 3.

---

[10] In this model the vehicle's weapons are assumed to be fully operational and all ammunition is assumed to be on board.

If the target is not firing at the user then the Target Side is requested with the responses being either friendly, enemy, or not_known. If friendly is chosen then the iteration ends. If enemy is chosen, the expert system enters the targeting phase mentioned before. The uncertainty addressed in the model comes into play if the user selects not_known as the response to Target Side.

When not_known is selected for Target Side then the user is shown a new set of questions. The questions and their responses are shown below.

Is the unknown target on, near or away from a road?:
On, Near, Away.

Is the target located near a known enemy or friendly position? Enemy, Friendly, Neither.

Are there more than one unknown targets? Yes, No.

Are the target(s) in formation? Yes, No.

Answers to these questions are run through "facts" similar to the status-check-function discussed earlier

in Chapter 8. The result is a value which is assigned to should-fire. If should-fire is yes with a cf greater than or equal to 80 then the following statement does the bulk of the work for the rest of the iteration.

```
if self-defense is sought and
        should-fire is sought and
        should-fire cf N and
        N >= 80 and
        weapon-selected is sought and
        ammunition-selected is sought and
        the-user-is-finished = Answer
then continue the-consultation = Answer.
```

If should-fire is less than 80, then this statement is executed.

```
if self-defense is sought and
    should-fire is sought and
    should-fire cf N and
    N < 80 and
    the-user-is-finished = Answer
then continue the-consultation = Answer.
```

Both of these statements are seeking the initial data expression continue the-consultation. If the-user-is-finished is yes (i.e., true) then continue the-consultation is set to true through the variable Answer. The opposite holds for a response of no (false). The final initial data element all-targets-engaged is then sought. However, all-targets-engaged is never reached because the expert gunnery system is always either restarted or aborted before it is.

To completely follow this model through several iterations please refer to Appendixes III and IV. M.1 has the ability to take a "photo" session of the screen as a user runs the program. M.1 also has the capability to show the user all of the rules as it uses them in search of goals. Appendix III is a session without a trace of all the rules. It is exactly what the normal user will see when running this expert system. Appendix IV is a session with the rule tracing element on.

The expression should-fire is the key to firing at an unknown target. The tuples impacting the certainty factor for should-fire are listed below.

The tuples for status-check-function are the first tuples evaluated. They are based on the friendly status and the enemy status in that order.

```
status-check-function(leading, attacking) = 60.
status-check-function(attacking, attacking) = 55.
status-check-function(defending, attacking) = 60.
status-check-function(in_reserve, attacking) = 50.
status-check-function(reconnaissance/observation, attacking) = 10.
status-check-function(withdrawing, attacking) = 50.
status-check-function(delay, attacking) = 50.

status-check-function(leading, defending) = 55.
status-check-function(attacking, defending) = 50.
status-check-function(defending, defending) = 40.
status-check-function(in_reserve, defending) = 30.
status-check-function(reconnaissance/observation, defending) = 5.
status-check-function(withdrawing, defending) = 5.
status-check-function(delay, defending) = 5.
```

```
status-check-function(leading, withdrawing) = 60.
status-check-function(attacking, withdrawing) = 55.
status-check-function(defending, withdrawing) = 30.
status-check-function(in_reserve, withdrawing) = 20.
status-check-function(reconnaissance/observation, withdrawing) = 10.
status-check-function(withdrawing, withdrawing) = 5.
status-check-function(delay, withdrawing) = 5.

status-check-function(leading, delay) = 60.
status-check-function(attacking, delay) = 55.
status-check-function(defending, delay) = 30.
status-check-function(in_reserve, delay) = 20.
status-check-function(reconnaissance/observation, delay) = 10.
status-check-function(withdrawing, delay) = 5.
status-check-function(delay, delay) = 5.
```

The tuples for dismounted-check-function are based on the target type, the presence of friendly patrols, and the presence of civilian refugees. As you can see, if the target type is infantry, dismounted, or uncertain, and either civilians or patrols are present, the certainty of should-fire is dramatically lower than if the target is any other type.

```
dismounted-check-function(infantry, no, no  ) =  10.
dismounted-check-function(infantry, no, yes ) = -10.
dismounted-check-function(infantry, yes, no ) = -20.
dismounted-check-function(infantry, yes, yes) = -30.

dismounted-check-function(dismounted, no, no  ) =  10 .
dismounted-check-function(dismounted, no, yes ) = -10.
dismounted-check-function(dismounted, yes, no ) = -20.
dismounted-check-function(dismounted, yes, yes) = -30.

dismounted-check-function(uncertain, no, no  ) =  10.
dismounted-check-function(uncertain, no, yes ) = -10.
dismounted-check-function(uncertain, yes, no ) = -20.
dismounted-check-function(uncertain, yes, yes) = -30.
```

```
dismounted-check-function(ANY, no, no  ) = 40.[11]
dismounted-check-function(ANY, no, yes ) = 30.
dismounted-check-function(ANY, yes, no ) = 20.
dismounted-check-function(ANY, yes, yes) = 10.
```

The tuples for location-check-function are based on whether or not the enemy is on, near, or away from a road, and if the target is near an enemy or friendly position. In principle, the further away from a road the target is, the more likely it is a combatant. Proximity to an enemy location will indicate the target is probably an enemy. Proximity to a friendly location implies friendly troops. Being close to neither can indicate nothing in some cases. These characteristics are shown in the certainty factors in the nine tuples for location-check-function. Note the low certainty factors associated with the last three tuples.

```
location-check-function(on  , enemy) = 20.
location-check-function(near, enemy) = 25.
location-check-function(away, enemy) = 30.

location-check-function(on  , friendly) = -20.
location-check-function(near, friendly) = -20.
location-check-function(away, friendly) = -20.
```

[11] **ANY** is a reserved word in M.1. It works for any value of an expression. In this case, if the target-type is not either dismounted, infantry, or unknown, then the four rules shown with ANY would apply. This lets the programmer escape the drudgery of writing out all of the tuples if there are only few exceptions to a general rule.

```
location-check-function(on   , neither) = 0.
location-check-function(near, neither) = 0.
location-check-function(away, neither) = 10.
```

The last tuples which determine should-fire when the target's side is unknown are formation-check-function. These tuples are based on whether or not there are multiple targets and also whether the multiple targets are in formation. A yes to both questions will increase the certainty factor for should-fire, as will a yes for multiple targets and a no for their being in formation. A "no-no" answer should obviously add nothing to the certainty factor, and a "no-yes" is an invalid answer. It is impossible not to have multiple targets (i.e., only one target) and have a formation.

```
formation-check-function(no,  no)  = 0.
formation-check-function(no,  yes) = 0.
formation-check-function(yes, no)  = 10.
formation-check-function(yes, yes) = 15.
```

The assignment of weapons and ammunition in the model is made with simple facts readily visible in Appendices I or II. These facts were made in accordance with Field Manual 23-1, Bradley Fighting Vehicle Gunnery, and accepted field artillery procedures for target engagement.

Validity of the Certainty Factors

In this model I have been both the knowledge engineer and the expert. This can be either good or bad depending on your point of view. The result is that the values for the certainty factors are based solely on my opinion. However, my opinions are no better or worse than any other infantry officer's. My experience has shown that very few officers would agree completely with the values I have given for the certainty factors. However, if I asked ten different Army officers to give me certainty factors for my model, I would receive ten different answers. Furthermore, these officers would not be able to justify their answers any better than I can. They would only "feel" that their answers were more accurate.

Despite all of the science and technology applied to "Military Science", some pieces of it are still an art; deciding the best time to act is one of these pieces. For these reasons I am satisfied with leaving the certainty factors as they are.

Model Validity

The Certainty Factor Theory developed by Buchanan and Shortliffe for the MYCIN system is imperfect. There have been many criticisms of the system and the theory behind it. Even Buchanan and Shortliffe agree that it is not a perfect system and that it has its flaws. [Buchanan and Shortliffe 84] Nevertheless, I chose this system over others for a variety of reasons. Some of these reasons are practical while others are academic.

First, I decided that I would not write an inference engine of my own. This could be a masters thesis topic in itself, which would have precluded any further analysis. This led me to use an existing commercial application.

Second, the existing commercial packages are very expensive. They typically cost several thousand dollars. Furthermore, most of the commercial packages do not deal with uncertainty. Many of those that do deal with uncertainty use MYCIN Certainty Factor theory. M.1 is one of these.

Third, after an extensive survey of all existing systems the U.S. Army (of which I am a member) decided to purchase the M.1 Inference Engine from Teknowledge. M.1 is currently used by the U.S. Army for prototyping small knowledge systems. I prefer this thesis to be of some use to the Army or the Defense Advanced Research Projects Agency. Writing it with M.1 will increase the likelihood of that occurring.

Fourth, I would like to see this thesis implemented. M.1 is written in "C" and has the ability to control external devices and drivers. My next assignment is as a computer science instructor at the United States Military Academy at West Point. I hope to have the time and the equipment to implement part of this thesis there.

All of these are practical considerations for using M.1. From an academic standpoint, I am disappointed that DARPA is so committed to using case-based reasoning for everything. Admittedly, case-based reasoning and machine learning will be great tools for the future. They are also the only technologies available to solve some problems (e.g., robotic vehicle drivers). But, limited domain problems like gunnery are solvable with

rule based systems. There is no need to invest the processing power or the memory requirements for a case-based reasoning system in a simple limited domain with a limited response.

DARPA is having some success with case-based reasoning for robotic vehicle "drivers". I believe the memory requirements for a robotic driver with case-based reasoning to be much less than those for a robotic commander with similar logic. This is apparent when you compare the experience of a vehicle commander with the experience of a driver. The commander is the most experienced member of the crew. Although the commander's decisions are controlled by only a few rules, the experience needed to gain them is important. This experience will require a lot of memory in a cased-based reasoning system.

However, the task of deciding whether or not to shoot at a target is a comparatively simple classification problem. Rule based systems are excellent for solving classification problems with low level input.

My goal in this thesis was to prove that a solution to a gunnery problem with uncertainty could be achieved with

a rule based system. It does not matter if that rule based system is implemented using certainty factors, fuzzy sets, or any other theoretical foundation. The fact remains that limited domain problems are solvable with rule based systems regardless of their complexity. I believe I have shown this in this thesis.

As far as the disadvantages of using certainty factors are concerned, I believe that I have successfully countered most of them. I tried very hard not to introduce any inconsistencies into the knowledge base. The problem with the length of the inference chain has been contended with by keeping the inference chain as short as possible. Finally, the "snapshot" nature of certainty factors is answered by having the system just look at "snapshots". The system can be used for extended periods because the system is only activated when the sensor detects a target. Thus, after making a decision, the system resets itself and waits for another target.

Chapter 10 Summary and Conclusions

I have explained the mental processes and the tools used in Bradley Infantry Fighting Vehicle gunnery.

First, I presented an overview of expert systems in general and the MYCIN certainty factor theory in particular. Next, I showed why an expert gunnery system would be a useful system.

Current gunnery methods and battle uncertainty were discussed in Chapters 4 and 5. I showed how gunnery is currently carried out and some of its inherent problems. Uncertainty in battle is very important. Mastering it is the key to success. Battlefield commanders rarely have a complete picture of the battle. They must guess. If expert systems that correctly make decisions in uncertain situations can be introduced at a low level on the battlefield, then men can be released for other duties.

Such expert systems would need information. I have discussed how these expert systems would get sensor information in Chapter 6.

The information available to the model is well within the reach of current sensor technology and map databases. The expert system is provided no more information than most vehicle commanders have available in combat.

Finally, I have discussed how to model a gunnery system and how to implement it with the M.1 inference engine.

The goal of this thesis is to prove that a rule based expert system for land gunnery is possible, and that it will work with uncertainty. The system I have constructed works reasonably well, and makes logical tactical choices from the information available to it.

In working towards this goal, I have achieved other benefits.

1) In this thesis I have presented a formal model for a gunnery system. This formal model breaks the gunnery process into discrete blocks. Each block builds on its predecessors until a decision is made.

2) This model may be of further use to the military in other problems or for analyzing the way we currently

teach soldiers to solve gunnery problems and battlefield situations.

3) I integrated very diverse resources into a comprehensible and meaningful thesis topic. This thesis combines my experience as an Infantry officer with what I have learned in my Computer Science studies. This combination allows me to look at some problems with a unique insight.

4) In developing this model, I have built a rule base for the expert system. This rule base combines my experience in both the Army and in Computer Science. The rule base contains 282 rules and implements the expert gunnery model as I have outlined in this thesis.

5) Finally, I have provided another proof of a MYCIN based expert system working in a classification problem. This application is new for a MYCIN based system.

The next task to be done on this system is to integrate sensors and databases into it. Portions of the program presented here could even be used. M.1 is written in "C" and can be made to call and control external "C"

routines and devices. I hope to continue this work at West Point during my tenure as an instructor.

BIBLIOGRAPHY

[Adams 76] Adams, J. B., A probability model of medical reasoning and the MYCIN model, Mathematical Biosciences, number 32, pp 177-186, 1976.

[Augarten 84] Augarten, S., Bit by Bit An Illustrated History of Computing, Ticknor & Fields, New York, 1984.

[BDM] BDM Corporation, Electromagnetic Spectrum, BDM Corporation, McLean, VA, 1985.

[Brownston 85] Brownston, L, Farrell, R., Kant, E., and Martin, N., Programming Expert Systems in OPS5, An Introduction to Rule-Based Programming, Addison-Wesley, Reading, Massachusets, 1985.

[Buchanan and Duda 83] Buchanan, B. and Duda, R., Principles of Rule-Based Expert Systems, Advances in Computers, edited by Yovits, M., pp. 163-216, 1983.

[Buchanan and Shortliffe 84] Buchanan, B. and Shortliffe, E., Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristics Programming Project, Addison-Wesley, Reading, Massachusets, 1984.

[Buchanan 86] Buchanan, B., Expert Systems: Working Systems and the Research Literature, Expert Systems, pp. 32-51, January 1986.

[Charniak and McDermott 87] Charniak, E. and McDermott, D., Introduction to Artificial Intelligence, Addison-Wesley, Reading, Massachusets, 1987.

[Cottrell 79] Cottrell, K., Try, P., Hodges, D., and Wachtmann, R., Electro-Optical Handbook Volume I, Weather Support for Precision Guided Munitions, Air Weather Service, Scott Air Force Base, Illinois, May 1979.

[Emmons 88] Telephone conversation with Dr George Emmons and Mr. Doug Crosswye, United States Army Material Command Smart Munitions Program Office, United States Army Missile Command, Redstone Arsenal, AL, May 1988.

[Felts] Felts, L., and Berg, F., Proposed Military Standardization Handbook Real Time Electro-Optical Imaging Systems, A Tutorial Guide to the Specification of Performance, Philco-Ford Corporation, undated.

[Grzymala-Busse 87] Grzymala-Busse, J., Class notes CS 790, Expert Systems, University of Kansas, 1987.

[Grzymala-Busse 88] Grzymala-Busse, J., Managing Uncertainty in Expert Systems, unpublished draft for text, University of Kansas, 1988.

[Harmon 85] Harmon, P., and King, D., Expert Systems, Artificial Intelligence in Business, John Wiley & Sons, New York, 1985.

[Jane's 87] Jane's Infantry Weapons, edited by Ian V. Hogg, pp. 758-759, 1987.

[Kolodner 88a] Kolodner, J., Extending Problem Solver Capabilities Through Case-Based Inference, Proceedings, DARPA Case-Based Reasoning Workshop, pp. 21-33, 10-13 May 1988.

[Kolodner 88b] Kolodner, J., Retrieving Events from a Case Memory: A Parallel Implementation, Proceedings, DARPA Case-Based Reasoning Workshop, pp. 233-249, 10-13 May 1988.

[Koton 88] Koton, P., Reasoning about Evidence in Causal Explanations, Proceedings, DARPA Case-Based Reasoning Workshop, pp. 260-270, 10-13 May 1988.

[Mich 83] Michalski, R, Carbonell, J., and Mitchell, T., Machine Learning, An Artificial Intelligence Approach, Tioga Publishing, Palo Alto, California, 1983.

[Rolston 88] Rolston, D., Principles of Artificial Intelligence and Expert Systems Development, McGraw-Hill Inc, New York, NY, 1988.

[Simpson 88] Telephone conversation with Lieutenant-Colonel Robert Simpson, Project Manager for Machine Intelligence in the Information Science and Technology Office, Defense Advanced Research Projects Agency, May 88.

[Staugaard 87] Staugaard, A., Robotics and AI, An Introduction to Applied Machine Intelligence, Prentice Hall, Englewood Cliffs, NJ, 1987.

[Teknowledge 86a] Teknowledge, Inc, M.1, Training Materials, Embarcaderas, California, April 1986.

[Teknowledge 86b] Teknowledge, Inc, M.1, Reference Manual for Software Version 2.1, Embarcaderas, California, December 1986.

[FM 6-20] United States Army Field Manual (FM) 6-20 Fire Support in Combined Arms Operations, United States Government Printing Office, 31 December 1984.

[FM 6-30] United States Army Field Manual (FM) 6-30 Observed Fire Procedures, United States Government Printing Office, 17 June 1985.

[FM 7-7J] United States Army Field Manual (FM) 7-7J The Mechanized Infantry Platoon and Squad (Bradley), United States Government Printing Office, 18 February 1986.

[FM 17-2] United States Army Field Manual (FM) 17-2 Tank Gunnery, United States Government Printing Office, 29 September 1978.

[FM 17-2-1] United States Army Field Manual (FM) 17-2-1 Tank Combat Tables, United States Government Printing Office, 17 July 1987.

[FM 23-1] United States Army Field Manual (FM) 23-1 Bradley Fighting Vehicle Gunnery, United States Government Printing Office, 30 September 1987.

[FM 23-34] United States Army Field Manual (FM) 23-34 TOW Weapon System, United States Government Printing Office, 30 September 1987.

[MICOMa] United States Army Material Command Smart Munitions Program Office, Smart Munitions: An Introduction to the Concepts, the Technologies, and the Systems Briefing Manual, United States Army Missile Command, Redstone Arsenal, AL, 29 September 1987.

[MICOMb] United States Army Material Command Smart Munitions Program Office, Smart Munitions: An Introduction to the Concepts, and the Technologies Primer, United States Army Missile Command, Redstone Arsenal, AL, 29 September 1987.

[Williams 85] Williams, M., A History of Computing Technology, Prentice-Hall, Englewood Cliffs, NJ, 1985.

APPENDIX I

Appendix I contains the source code for the expert gunnery system.

```
/*
Bradley Fighting Vehicle Expert Gunner
Version 1

Date: 17 August 1988

Author:   Timothy J. Gibson
            Captain, Infantry
            U.S. Army

This knowledge system has been built using M.1 (c) Expert System Shell
version 2.1(1) from

     Teknowledge, Inc.
     1850 Embarcadera Road
     Palo Alto, CA  94303

An original manufacturer's copy of M.1 (S/N 06791) was provided to the author by the
Department of Computer Science at the United States Military Academy, West Point, N.Y.
10996-1695 for use solely on this Master's Thesis at the University of Kansas, Lawrence, KS
66045.

*/
prefix begin.
prefix end.
prefix continue.
initialdata =
[begin the-consultation, continue the-consultation, all-targets-engaged].

/*
Header
*/
begin message = [
nl," ",
nl,"         The M2/M3 Bradley Fighting Vehicle Gunner Advisor ",
nl," ",
nl,"         A system to assist a vehicle commander in deciding ",
nl,"         whether or not to fire on known or unknown targets,",
nl,"         and to provide the commander with the correct weapon",
nl,"         and ammunition combination for the job.            ",
nl,"                                                            ",

nl,"            Built by CPT Timothy J. Gibson."].
```

```
nocache(begin message).

question(begin signal) = [
nl," In order to run this weapon adviser the user needs to know",
nl," or have access to the range to specific targets or a rangefinder,",
nl," a map, and current intelligence reports.",
nl,"  ",
nl,"  ",
nl,"          Are you prepared at this time?"].
legalvals(begin signal) = [yes, no].
automaticmenu(begin signal).
enumeratedanswers(begin signal).



/*
This section of code displays the opening message and asks the user if
he/she is ready to begin.  If they are ready to start then the program
asks the user for some background information.
*/


if begin message = M and
          display(M) and
          begin signal and
          display("\f") and
          friendly-status is sought and
          enemy-status is sought and
          enemy-unit-type is sought and
          patrol-status is sought and
          civilian-status is sought and
          display("\f")
then begin the-consultation.

if (not begin signal) and
     display("\t Tell the computer 'go' when you are ready to proceed \n") and
     display("\t or 'exit' to leave the M.1 shell.") and
     do(abort)
     then begin the-consultation.
```

```
/*
Background Questions
on the status of friendly and enemy forces,
friendly air defense status, friendly patrol activity,
and non-combatants.
*/
explanation(friendly-determination) = [
"Answering this question provides needed background information for \n",
"the gunnery model.\n"].
question(friendly-determination) =
["\n \t Background Questions \n\n\tWhat is your units' current tactical posture?\n"].
legalvals(friendly-determination) =
[attacking,defending,in_reserve,reconnaissance/observation,withdrawing,delay].
automaticmenu(friendly-determination).
enumeratedanswers(friendly-determination).

explanation(friendly-leading) = [
"Answering this question provides needed background information for \n",
"the gunnery model.\n"].
question(friendly-leading) =
["\n \t Is your unit leading the attack?\n"].
legalvals(friendly-leading) = [yes, no].
automaticmenu(friendly-leading).
enumeratedanswers(friendly-leading).

if friendly-determination = attacking and
friendly-leading = yes
then friendly-status = leading.

if friendly-determination = Anything_else
then friendly-status = Anything_else.


explanation(enemy-status) = [
"Answering this question provides needed background information for \n",
"the gunnery model.\n"].
question(enemy-status) =
["\f \n\t Background Questions\n\n\tWhat is enemy's current tactical posture?\n"].
legalvals(enemy-status) = [attacking,defending,withdrawing,delay].
automaticmenu(enemy-status).
enumeratedanswers(enemy-status).
```

```
explanation(enemy-unit-type) = [
"Answering this question provides needed background information for \n",
"the gunnery model.\n"].
question(enemy-unit-type) =
["\n \t What type of enemy unit are you currently facing \n"].
legalvals(enemy-unit-type) = [tank, apc, infantry].
automaticmenu(enemy-unit-type).
enumeratedanswers(enemy-unit-type).


explanation(patrol-status) = [
"Answering this question provides needed background information for \n",
"the gunnery model.\n"].
question(patrol-status) =
["\f \n\tBackground Questions\n\n",
"\tAre there friendly patrols operating in this area? \n"].
legalvals(patrol-status) = [yes, no].
automaticmenu(patrol-status).
enumeratedanswers(patrol-status).

explanation(civilian-status) = [
"Answering this question provides needed background information for \n",
"the gunnery model.\n"].
question(civilian-status) =
["\n \t Are there civilian refugees in this area? \n"].
legalvals(civilian-status) = [yes, no].
automaticmenu(civilian-status).
enumeratedanswers(civilian-status).



/*
This code section sets the expression should-fire to no with a cf of 100, tells the user not
to shoot at confirmed friendly troops and prompts the user for a continuation.
*/
if should-fire = no and
         display("\f \n \n \n \t Do not shoot at friendly troops.\n \n ") and
         the-user-is-finished = Answer
then continue the-consultation = Answer.
```

```
/*
This code section sets the expression should-fire to yes with a cf of 100 if the friendly
unit is being fired on by the target.
*/

if self-defense  = yes and
         display("\n \t Firing in self defense is always authorized. \n")
then should-fire = yes cf 100.

/*
This code block checks for a should-fire certainty of 80% or higher and then moves into the
targeting procedure.  The selection of weapons and ammunition by the program concludes in a
message to the user and a question about continuing.
*/

if self-defense is sought and
         should-fire is sought and
         should-fire cf N and
         N >= 80 and
         weapon-selected is sought and
         ammunition-selected is sought and
         output-range and
         output-weapon and
         the-user-is-finished = Answer
then continue the-consultation = Answer.

/*
This code block checks for a should-fire certainty of 80% or less and then tells the user why
firing is not advised.
*/

if self-defense is sought and
         should-fire is sought and
         should-fire cf N and
         N < 80 and
         too-low-patrol and
         too-low-civilian and
         display("\f") and
         display("The probability of the target being an enemy is \n") and
         display("too low to justify engaging it.\n") and
         the-user-is-finished = Answer
then continue the-consultation = Answer.

/*
If the user wants to quit, the program terminates.
*/
```

```
if continue the-consultation = no and
         do(abort)
then all-targets-engaged.

/*
If the user wants to continue, the program resets the necessary expressions and restarts
itself.
*/
if continue the-consultation and
         do(reset continue the-consultation) and
         do(reset self-defense) and
         do(reset should-fire ) and
         do(reset the-target) and
         do(reset the-side) and
         do(reset the-terrain) and
         do(reset the-range) and
         do(reset weapon-selected) and
         do(reset ammunition-selected) and
         do(reset maybe-should-fire) and
         do(reset dismounted-check) and
         do(reset location-check) and
         do(reset formation-check) and
         do(reset unknown-near-road) and
         do(reset unknown-location) and
         do(reset unknown-multiple-target) and
         do(reset output-range) and
         do(reset output-weapon) and
         do(reset unknown-in-formation) and
         do(reset the-user-is-finished )and
         do(restart)
then all-targets-engaged.

/*********
********** Facts for correct output.
**********/
if patrol-status and
    display("There is a possibility that the target is a friendly patrol.\n")
then too-low-patrol.

if not patrol-status
then too-low-patrol.

if civilian-status and
    display("There is a possibility that the target is a party of noncombatants.\n")
then too-low-civilian.
```

```
if not civilian-status
then too-low-civilian.

if mostlikely(the-range) = Range and
          display("\f \n \n \n \t The range to the target is ") and
          display(Range) and
          display(" meters.\n \n \n")
then output-range.

if mostlikely(weapon-selected) = Weapon and
     Weapon == indirect and
     mostlikely(ammunition-selected) = Ammunition and
     display("\t The preferred weapon is artillery fire because \n") and
     display("\t of the long distance to the target.\n  \n") and
     display("\t Request indirect fire through your fire support channels.") and
     display("\n \t The preferred munition is ") and
     display(Ammunition)
then output-weapon.

if mostlikely(weapon-selected) = Weapon and
          mostlikely(ammunition-selected) = Ammunition and
          display("\t The preferred weapon is the ") and
          display(Weapon) and
          display("\n \t firing ") and
          display(Ammunition) and
          display(" ammunition.")
then output-weapon.


/*
Ask about self defense
*/
explanation(self-defense) = [
"\n Firing in self defense is always allowed.  Answering\n",
"yes to this question will take the user to the targeting menus\n",
"immediately.\n"].
question(self-defense) =
["\f \n \n \t \t Is the target currently firing at you? \n \n"].
legalvals(self-defense) = [yes, no].
automaticmenu(self-defense).
enumeratedanswers(self-defense).
```

```
/*
** Determine if the user is finished.
*/

question(the-user-is-finished) =
["\n \t This concludes one iteration of the Bradley Gunner.  Do\n ",
"\t you wish to continue?\n "].
legalvals(the-user-is-finished) = [yes, no].
automaticmenu(the-user-is-finished).
enumeratedanswers(the-user-is-finished).



/*
*******
********      Ask the user which side the target is on.
*******
*/

/*
** Set should-fire to yes if the user is being fired on.
*/
if the-side = enemy
then should-fire = yes.

/*
** Set should-fire to no if the target is a known friendly.
*/
if the-side is sought and
         the-side = friendly
then should-fire = no.
```

```
/*
If the side of the enemy is not known the program will evaluate the background information
given at the beginning of the program and combine it with some additional information asked
of the user to determine whether or not to shoot at the target.
*/
if the-side is sought and
          the-side = not_known and
          status-check = SC and
          do(set maybe-should-fire = yes cf SC) and
          dismounted-check = DC and
          do(set maybe-should-fire = yes cf DC) and
          location-check = LC and
          do(set maybe-should-fire = yes cf LC) and
          formation-check = FC and
          do(set maybe-should-fire = yes cf FC) and
          maybe-should-fire = Final
then should-fire = Final.

/*
** Function call to status-check-function
*/
if friendly-status = FS and
          enemy-status = ES and
          status-check-function(FS, ES) = SCF
then status-check = SCF.

status-check-function(leading, attacking) = 60.
status-check-function(attacking, attacking) = 55.
status-check-function(defending, attacking) = 60.
status-check-function(in_reserve, attacking) = 50.
status-check-function(reconnaissance/observation, attacking) = 10.
status-check-function(withdrawing, attacking) = 50.
status-check-function(delay, attacking) = 50.
```

```
status-check-function(leading, defending) = 55.
status-check-function(attacking, defending) = 50.
status-check-function(defending, defending) = 40.
status-check-function(in_reserve, defending) = 30.
status-check-function(reconnaissance/observation, defending) = 5.
status-check-function(withdrawing, defending) = 5.
status-check-function(delay, defending) = 5.

status-check-function(leading, withdrawing) = 60.
status-check-function(attacking, withdrawing) = 55.
status-check-function(defending, withdrawing) = 30.
status-check-function(in_reserve, withdrawing) = 20.
status-check-function(reconnaissance/observation, withdrawing) = 10.
status-check-function(withdrawing, withdrawing) = 5.
status-check-function(delay, withdrawing) = 5.

status-check-function(leading, delay) = 60.
status-check-function(attacking, delay) = 55.
status-check-function(defending, delay) = 30.
status-check-function(in_reserve, delay) = 20.
status-check-function(reconnaissance/observation, delay) = 10.
status-check-function(withdrawing, delay) = 5.
status-check-function(delay, delay) = 5.


/*
** Function call to dismounted-check-function
*/
if the-target = UTT and
          patrol-status = PS and
          civilian-status = CS and
          dismounted-check-function(UTT, PS, CS) = DCF
then dismounted-check = DCF.


dismounted-check-function(infantry, no, no  ) =  10.
dismounted-check-function(infantry, no, yes ) = -10.
dismounted-check-function(infantry, yes, no ) = -20.
dismounted-check-function(infantry, yes, yes) = -30.

dismounted-check-function(dismounted, no, no  ) =  10 .
dismounted-check-function(dismounted, no, yes ) = -10.
dismounted-check-function(dismounted, yes, no ) = -20.
dismounted-check-function(dismounted, yes, yes) = -30.
```

```
dismounted-check-function(uncertain, no, no  ) =  10 .
dismounted-check-function(uncertain, no, yes ) = -10.
dismounted-check-function(uncertain, yes, no ) = -20.
dismounted-check-function(uncertain, yes, yes) = -30.

dismounted-check-function(ANY, no, no  ) = 40.
dismounted-check-function(ANY, no, yes ) = 30.
dismounted-check-function(ANY, yes, no ) = 20.
dismounted-check-function(ANY, yes, yes) = 10.

explanation(the-target) = [
"Different types of targets require different type of weapons and \n",
"ammunitions to destroy them.  Answering this question will let the \n",
"gunner choose the best weapon and ammunition combination."].
question(the-target) = ("\f \n \t What type of target is it? \n ").
legalvals(the-target) =
[tank, apc, sp_artillery, towed_artillery, anti_tank_unit, infantry, trucks, logistics_area,
command_post, tracked, wheeled, dismounted, uncertain].
automaticmenu(the-target).
enumeratedanswers(the-target).


/*
** Function call to location-check-function
*/
if unknown-near-road = UNR and
          unknown-location = UL and
          location-check-function(UNR, UL) = LCF
then location-check = LCF.

explanation(unknown-near-road) = [
"You have stated that the identity of the target is not known. \n",
"This question will assist the gunner in determining whether \n",
"or not to fire at the target."].
question(unknown-near-road) =
("\f \n \t Is the unknown target on, near or away from a road?\n ").
legalvals(unknown-near-road) = [on, near, away].
automaticmenu(unknown-near-road).
enumeratedanswers(unknown-near-road).
```

```
explanation(unknown-location) = [
"You have stated that the identity of the target is not known. \n",
"This question will assist the gunner in determining whether \n",
"or not to fire at the target."].
question(unknown-location) =
("\n \t Is the target near a known enemy or friendly location? \n ").
legalvals(unknown-location) = [enemy, friendly, neither].
automaticmenu(unknown-location).
enumeratedanswers(unknown-location).


location-check-function(on   , enemy) = 20.
location-check-function(near, enemy) = 25.
location-check-function(away, enemy) = 30.

location-check-function(on   , friendly) = -20.
location-check-function(near, friendly) = -20.
location-check-function(away, friendly) = -20.

location-check-function(on   , neither) = 0.
location-check-function(near, neither) = 0.
location-check-function(away, neither) = 10.



/*
** Function call to formation-check-function
*/
if unknown-multiple-target = UMT and
          unknown-in-formation = UF and
          formation-check-function(UMT, UF) = FCF
then formation-check = FCF.


explanation(unknown-multiple-target) = [
"You have stated that the identity of the target is not known. \n",
"This question will assist the gunner in determining whether \n",
"or not to fire at the target."].
question(unknown-multiple-target) =
("\f \n Are there more than one unknown targets? \n ").
legalvals(unknown-multiple-target) = [yes, no].
automaticmenu(unknown-multiple-target).
enumeratedanswers(unknown-multiple-target).
```

```
explanation(unknown-in-formation) = [
"You have stated that the identity of the target is not known. \n",
"This question will assist the gunner in determining whether \n",
"or not to fire at the target."].
question(unknown-in-formation) = ("\nIs the target(s) in a formation?\n ").
legalvals(unknown-in-formation) = [yes, no].
automaticmenu(unknown-in-formation).
enumeratedanswers(unknown-in-formation).

formation-check-function(no,  no)  = 0.
formation-check-function(no,  yes) = 0.
formation-check-function(yes, no)  = 10.
formation-check-function(yes, yes) = 15.



explanation(the-side) = [
"Some type of determination must be made. \n ",
"You must answer this question to continue.\n"].
question(the-side) = "\n \tIs the target enemy, friendly, or not known?\n".
legalvals(the-side) = [enemy, friendly, not_known].
automaticmenu(the-side).
enumeratedanswers(the-side).



/*
The following section contain rules for selecting the correct weapon and shell combination
for the proper target.
*/

/*
**  Selecting weapon types
*/
```

```
/*  Tanks */

if the-target = tank
                or
          (the-target = uncertain and enemy-unit-type = tank)
                or
          (the-target = tracked and enemy-unit-type = tank)
                or
          (the-target = tracked and enemy-unit-type = infantry)
                and
          the-range = Range and
          Range >= 3751 and
          weapon-type(tank, extreme) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = tank
                or
          (the-target = uncertain and enemy-unit-type = tank)
                or
          (the-target = tracked and enemy-unit-type = tank)
                or
          (the-target = tracked and enemy-unit-type = infantry)
                and
          the-range = Range and
          Range < 3751 and
          Range >= 65 and
          weapon-type(tank, long) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = tank
                or
          (the-target = uncertain and enemy-unit-type = tank)
                or
          (the-target = tracked and enemy-unit-type = tank)
                or
          (the-target = tracked and enemy-unit-type = infantry)
                and
          the-range = Range and
          Range < 65 and
          weapon-type(tank, close) = Weapon_return
then weapon-selected = Weapon_return.
```

```
/* Armored Personnel Carriers */

if the-target = apc
                or
          (the-target = uncertain and enemy-unit-type = apc)
                or
          (the-target = tracked and enemy-unit-type = apc)
                and
          the-range = Range and
          Range >= 3751 and
          weapon-type(apc, extreme) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = apc
                or
          (the-target = uncertain and enemy-unit-type = apc)
                or
          (the-target = tracked and enemy-unit-type = apc)
                and
          the-range = Range and
          Range < 3751 and
          Range >= 1700 and
          weapon-type(apc, long) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = apc
                or
          (the-target = uncertain and enemy-unit-type = apc)
                or
          (the-target = tracked and enemy-unit-type = apc)
                and
          the-range = Range and
          Range < 1700 and
          weapon-type(apc, medium) = Weapon_return
then weapon-selected = Weapon_return.

/* Self-propelled Artillery */

if the-target = sp_artillery and
          the-range = Range and
          Range >= 3751 and
          weapon-type(apc, extreme) = Weapon_return
then weapon-selected = Weapon_return.
```

```
if the-target = sp_artillery and
          the-range = Range and
          Range < 3751 and
          Range > 1700 and
          weapon-type(apc, long) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = sp_artillery and
          the-range = Range and
          Range <= 1700 and
          weapon-type(apc, medium) = Weapon_return
then weapon-selected = Weapon_return.

/* Towed Artillery */
if the-target = towed_artillery and
          the-range = Range and
          Range > 3000 and
          weapon-type(equipment, long) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = towed_artillery and
          the-range = Range and
          Range <= 3000 and
          weapon-type(equipment, medium) = Weapon_return
then weapon-selected = Weapon_return.

/* Anti-Tank Units */

if the-target = anti_tank_unit and
          the-range = Range and
          Range > 3000 and
          weapon-type(equipment, long) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = anti_tank_unit and
          the-range = Range and
          Range <= 3000 and
          weapon-type(equipment, medium) = Weapon_return
then weapon-selected = Weapon_return.
```

```
/* Infantry */
if the-target = infantry
                or
          (the-target = uncertain and enemy-unit-type = infantry)
                or
          (the-target = dismounted and enemy-unit-type = infantry)
                and
          the-range = Range and
          Range >= 3000 and
          weapon-type(infantry, long) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = infantry
                or
          (the-target = uncertain and enemy-unit-type = infantry)
                or
          (the-target = dismounted and enemy-unit-type = infantry)
                and
          the-range = Range and
          Range < 3000 and
          Range >= 800 and
          weapon-type(infantry, medium) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = infantry
                or
          (the-target = uncertain and enemy-unit-type = infantry)
                or
          (the-target = dismounted and enemy-unit-type = infantry)
                and
          the-range = Range and
          Range <= 800 and
          weapon-type(infantry, close) = Weapon_return
then weapon-selected = Weapon_return.

/* Trucks */
if (the-target = trucks ) or (the-target = wheeled)
                and
          the-range = Range and
          Range >= 3000 and
          weapon-type(infantry, long) = Weapon_return
then weapon-selected = Weapon_return.
```

```
if (the-target = trucks ) or (the-target = wheeled)
                and
          the-range = Range and
          Range < 3000 and
          Range >= 800 and
          weapon-type(infantry, medium) = Weapon_return
then weapon-selected = Weapon_return.

if (the-target = trucks ) or (the-target = wheeled)
                and
          the-range = Range and
          Range <= 800 and
          weapon-type(infantry, close) = Weapon_return
then weapon-selected = Weapon_return.

/* Logistics Areas */
if the-target = logistics_area and
          the-range = Range and
          Range >= 3000 and
          weapon-type(cp, long) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = logistics_area and
          the-range = Range and
          Range < 3000 and
          Range > 100 and
          weapon-type(cp, medium) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = logistics_area and
          the-range = Range and
          Range <= 100 and
          weapon-type(cp, close) = Weapon_return
then weapon-selected = Weapon_return.

/* Command Posts */
if the-target = command_post and
          the-range = Range and
          Range >= 3000 and
          weapon-type(cp, long) = Weapon_return
then weapon-selected = Weapon_return.
```

```
if the-target = command_post and
          the-range = Range and
          Range < 3000 and
          Range > 100 and
          weapon-type(cp, medium) = Weapon_return
then weapon-selected = Weapon_return.

if the-target = command_post and
          the-range = Range and
          Range <= 100 and
          weapon-type(cp, close) = Weapon_return
then weapon-selected = Weapon_return.


explanation(the-range) = ["This is required data.  Guess if necessary.\n"].
question(the-range) = ("\n \t What is the range to the target in meters?").
legalvals(the-range) = number.


explanation(the-terrain) = ["Indirect fire has been chosen as the best weapon.\n",
"What the terrain is like will influence the preferred shell and\n",
"shell and fuze combination.\n"].
question(the-terrain) = ("\n \t What type of terrain is the target in? \n ").
legalvals(the-terrain) = [open, wooded, trenches/dug_in, town].
automaticmenu(the-terrain).
enumeratedanswers(the-terrain).

weapon-type(tank, extreme) = indirect.
weapon-type(tank, long) = guided_missile.
weapon-type(tank, medium) = guided_missile.
weapon-type(tank, close) = automatic_cannon.

weapon-type(apc, extreme) = indirect.
weapon-type(apc, long) = guided_missile.
weapon-type(apc, medium) = automatic_cannon.
weapon-type(apc, close) = automatic_cannon.

weapon-type(equipment, extreme) = indirect.
weapon-type(equipment, long) = indirect.
weapon-type(equipment, medium) = automatic_cannon.
weapon-type(equipment, close) = automatic_cannon.

weapon-type(infantry, extreme) = indirect.
weapon-type(infantry, long) = indirect.
weapon-type(infantry, medium) = automatic_cannon.
weapon-type(infantry, close) = machinegun.

weapon-type(cp, extreme) = indirect.
weapon-type(cp, long) = indirect.
weapon-type(cp, medium) = automatic_cannon.
```

```
weapon-type(cp, close) = machinegun.



/*
Selecting ammunition types
*/
if weapon-selected = guided_missile
then ammunition-selected = anti_tank.

if weapon-selected = automatic_cannon and
          the-target = uncertain and
          enemy-unit-type = R and
          ammunition-type(automatic_cannon, R) = Ammunition_return
then ammunition-selected = Ammunition_return.

if weapon-selected = automatic_cannon and
          the-target = R and
          ammunition-type(automatic_cannon, R) = Ammunition_return
then ammunition-selected = Ammunition_return.

if weapon-selected = machinegun
then ammunition-selected = ball/tracer.

if weapon-selected = indirect and
          the-terrain is sought and
          the-terrain = T and
          the-target = uncertain and
          enemy-unit-type = R and
          artillery-type(T, R) = Artillery_return
then ammunition-selected = Artillery_return.

if weapon-selected = indirect and
          the-terrain is sought and
          the-terrain = T and
          the-target = R and
          artillery-type(T, R) = Artillery_return
then ammunition-selected = Artillery_return.
```

```
ammunition-type(automatic_cannon, tank) = armor_piercing_discarding_sabot.
ammunition-type(automatic_cannon, apc) = armor_piercing_discarding_sabot.
ammunition-type(automatic_cannon, sp_artillery) = armor_piercing_discarding_sabot.
ammunition-type(automatic_cannon, towed_artillery) = high_explosive_incendiary.
ammunition-type(automatic_cannon, anti_tank_unit) = high_explosive_incendiary.
ammunition-type(automatic_cannon, infantry) = high_explosive_incendiary.
ammunition-type(automatic_cannon, trucks) = high_explosive_incendiary.
ammunition-type(automatic_cannon, logistics_area ) = high_explosive_incendiary.
ammunition-type(automatic_cannon, command_post) = high_explosive_incendiary.
ammunition-type(automatic_cannon, uncertain) = high_explosive_incendiary.
ammunition-type(automatic_cannon, tracked) = armor_piercing_discarding_sabot.
ammunition-type(automatic_cannon, wheeled) = high_explosive_incendiary.
ammunition-type(automatic_cannon, dismounted) = high_explosive_incendiary.
ammunition-type(automatic_cannon, uncertain) = high_explosive_incendiary.

/*
Indirect fire options
*/
artillery-type(open,tank ) = dual_purpose_improved_conventional_munition.
artillery-type(open,apc ) = dual_purpose_improved_conventional_munition.
artillery-type(open,sp_artillery ) = dual_purpose_improved_conventional_munition.
artillery-type(open,towed_artillery ) = high_explosive_vt.
artillery-type(open,anti_tank_unit ) = high_explosive_vt.
artillery-type(open,infantry ) = high_explosive_vt.
artillery-type(open,trucks ) = high_explosive_vt.
artillery-type(open,logistics_area ) = high_explosive_and_white_phosphorus.
artillery-type(open,command_post ) = high_explosive_and_white_phosphorus.
artillery-type(open,tracked) = dual_purpose_improved_conventional_munition.
artillery-type(open,wheeled) = high_explosive_vt.
artillery-type(open,dismounted) = high_explosive_vt.
artillery-type(open,uncertain ) = high_explosive.

artillery-type(wooded, tank) = high_explosive_delay.
artillery-type(wooded, apc) = high_explosive_delay.
artillery-type(wooded, sp_artillery) = high_explosive_delay.
artillery-type(wooded, towed_artillery) = high_explosive_delay.
artillery-type(wooded, anti_tank_unit) = high_explosive_delay.
artillery-type(wooded, infantry) = high_explosive_delay.
artillery-type(wooded, trucks) = high_explosive_delay.
artillery-type(wooded, logistics_area) = high_explosive_delay_and_white_phosphorus.
artillery-type(wooded, command_post) = high_explosive_delay_and_white_phosphorus.
artillery-type(wooded, tracked) = high_explosive_delay.
artillery-type(wooded, wheeled) = high_explosive_delay.
artillery-type(wooded, dismounted) = high_explosive_delay.
artillery-type(wooded, uncertain) = high_explosive_delay.
```

```
artillery-type(trenches/dug_in, tank) = dual_purpose_improved_conventional_munition.
artillery-type(trenches/dug_in, apc) = dual_purpose_improved_conventional_munition.
artillery-type(trenches/dug_in, sp_artillery) = dual_purpose_improved_conventional_munition.
artillery-type(trenches/dug_in, towed_artillery) =
high_explosive_vt_and_high_explosive_delay.
artillery-type(trenches/dug_in, anti_tank_unit) = high_explosive_vt_and_high_explosive_delay.
artillery-type(trenches/dug_in, infantry) = high_explosive_vt_and_high_explosive_delay.
artillery-type(trenches/dug_in, trucks) = high_explosive_vt.
artillery-type(trenches/dug_in, logistics_area) = high_explosive_vt_and_high_explosive_delay.
artillery-type(trenches/dug_in, command_post) = high_explosive_vt_and_high_explosive_delay.
artillery-type(trenches/dug_in, tracked) = dual_purpose_improved_conventional_munition.
artillery-type(trenches/dug_in, wheeled) = high_explosive_vt_and_high_explosive_delay.
artillery-type(trenches/dug_in, dismounted) = high_explosive_vt_and_high_explosive_delay.
artillery-type(trenches/dug_in, uncertain) = high_explosive_vt_and_high_explosive_delay.

artillery-type(town, tank) =
dual_purpose_improved_conventional_munition_and_high_explosive_delay.
artillery-type(town, apc) =
dual_purpose_improved_conventional_munition_and_high_explosive_delay.
artillery-type(town, sp_artillery) =
dual_purpose_improved_conventional_munition_and_high_explosive_delay.
artillery-type(town, towed_artillery) = high_explosive_and_high_explosive_delay.
artillery-type(town, anti_tank_unit) = high_explosive_and_high_explosive_delay.
artillery-type(town, infantry) = high_explosive_and_high_explosive_delay.
artillery-type(town, trucks) = high_explosive_and_high_explosive_delay.
artillery-type(town, logistics_area) = high_explosive_and_white_phosphorus.
artillery-type(town, command_post) = high_explosive_and_high_explosive_delay.
artillery-type(town, tracked) =
dual_purpose_improved_conventional_munition_and_high_explosive_delay.
artillery-type(town, wheeled) = high_explosive_and_high_explosive_delay.
artillery-type(town, dismounted) = high_explosive_and_high_explosive_delay.
artillery-type(town, uncertain) = high_explosive_and_high_explosive_delay.
```

APPENDIX II

Appendix II contains the rules for the expert gunnery system as they are represented in the computer. Appendix V refers to the rules with the knowledge base numbers presented here.(e.g., kb-1 is the initialdata set).

```
kb-1:
    initialdata = [begin the-consultation,continue the-consultation,all-
    targets-engaged].

kb-2:
    begin message = [nl,' ',nl,
    '         The M2/M3 Bradley Fighting Vehicle Gunner Advisor ',nl,' ',nl,
    '         A system to assist a vehicle commander in deciding ',nl,
    '         whether or not to fire on known or unknown targets,',nl,
    '         and to provide the commander with the correct weapon',nl,
    '         and ammunition combination for the job.             ',nl,
    '                                                             ',nl,
    '            Built by CPT Timothy J. Gibson.'].

kb-3:
    nocache(begin message).

kb-4:
    question(begin signal) = [nl,
    ' In order to run this weapon adviser the user needs to know',nl,
    ' or have access to the range to specific targets or a rangefinder,',nl,
    ' a map, and current intelligence reports.',nl,'  ',nl,'  ',nl,
    '           Are you prepared at this time?'].

kb-5:
    legalvals(begin signal) = [yes,no].

kb-6:
    automaticmenu(begin signal).

kb-7:
    enumeratedanswers(begin signal).
```

```
kb-8:
    if begin message = M and
       display(M) and
       begin signal and
       display('\f') and
       friendly-status is sought and
       enemy-status is sought and
       enemy-unit-type is sought and
       patrol-status is sought and
       civilian-status is sought and
       display('\f')
    then begin the-consultation.

kb-9:
    if not begin signal and
       display('\t Tell the computer ''go'' when you are ready to proceed \n')
        and
       display('\t or ''exit'' to leave the M.1 shell.') and
       do(abort)
    then begin the-consultation.

kb-10:
    explanation(friendly-determination) =
    ['Answering this question provides needed background information for \n',
    'the gunnery model.\n'].

kb-11:
    question(friendly-determination) =
    ['\n \t Background Questions \n\n\tWhat is your units'' current tactical posture?\n']
    .

kb-12:
    legalvals(friendly-determination) = [attacking,defending,in_reserve,
    reconnaissance/observation,withdrawing,delay].

kb-13:
    automaticmenu(friendly-determination).

kb-14:
    enumeratedanswers(friendly-determination).

kb-15:
    explanation(friendly-leading) =
    ['Answering this question provides needed background information for \n',
    'the gunnery model.\n'].
```

```
kb-16:
    question(friendly-leading) = ['\n \t Is your unit leading the attack?\n'].

kb-17:
    legalvals(friendly-leading) = [yes,no].

kb-18:
    automaticmenu(friendly-leading).

kb-19:
    enumeratedanswers(friendly-leading).

kb-20:
    if friendly-determination = attacking and
       friendly-leading = yes
    then friendly-status = leading.

kb-21:
    if friendly-determination = Anything_else
    then friendly-status = Anything_else.

kb-22:
    explanation(enemy-status) =
    ['Answering this question provides needed background information for \n',
    'the gunnery model.\n'].

kb-23:
    question(enemy-status) =
    ['\f \n\t Background Questions\n\n\tWhat is enemy''s current tactical posture?\n']
    .

kb-24:
    legalvals(enemy-status) = [attacking,defending,withdrawing,delay].

kb-25:
    automaticmenu(enemy-status).

kb-26:
    enumeratedanswers(enemy-status).

kb-27:
    explanation(enemy-unit-type) =
    ['Answering this question provides needed background information for \n',
    'the gunnery model.\n'].
```

```
kb-28:
    question(enemy-unit-type) =
    ['\n \t What type of enemy unit are you currently facing \n'].

kb-29:
    legalvals(enemy-unit-type) = [tank,apc,infantry].

kb-30:
    automaticmenu(enemy-unit-type).

kb-31:
    enumeratedanswers(enemy-unit-type).

kb-32:
    explanation(patrol-status) =
    ['Answering this question provides needed background information for \n',
    'the gunnery model.\n'].

kb-33:
    question(patrol-status) = ['\f \n\tBackground Questions\n\n',
    '\tAre there friendly patrols operating in this area? \n'].

kb-34:
    legalvals(patrol-status) = [yes,no].

kb-35:
    automaticmenu(patrol-status).

kb-36:
    enumeratedanswers(patrol-status).

kb-37:
    explanation(civilian-status) =
    ['Answering this question provides needed background information for \n',
    'the gunnery model.\n'].

kb-38:
    question(civilian-status) =
    ['\n \t Are there civilian refugees in this area? \n'].

kb-39:
    legalvals(civilian-status) = [yes,no].

kb-40:
    automaticmenu(civilian-status).
```

```
kb-41:
    enumeratedanswers(civilian-status).

kb-42:
    if should-fire = no and
       display('\f \n \n \n \t Do not shoot at friendly troops.\n \n ') and
       the-user-is-finished = Answer
    then continue the-consultation = Answer.

kb-43:
    if self-defense = yes and
       display('\n \t Firing in self defense is always authorized. \n')
    then should-fire = yes cf 100.

kb-44:
    if self-defense is sought and
       should-fire is sought and
       should-fire cf N and
       N>= 80 and
       weapon-selected is sought and
       ammunition-selected is sought and
       output-range and
       output-weapon and
       the-user-is-finished = Answer
    then continue the-consultation = Answer.

kb-45:
    if self-defense is sought and
       should-fire is sought and
       should-fire cf N and
       N<80 and
       too-low-patrol and
       too-low-civilian and
       display('\f') and
       display('The probability of the target being an enemy is \n') and
       display('too low to justify engaging it.\n') and
       the-user-is-finished = Answer
    then continue the-consultation = Answer.

kb-46:
    if continue the-consultation = no and
       do(abort)
    then all-targets-engaged.
```

```
kb-47:
    if continue the-consultation and
       do(reset continue the-consultation) and
       do(reset self-defense) and
       do(reset should-fire) and
       do(reset the-target) and
       do(reset the-side) and
       do(reset the-terrain) and
       do(reset the-range) and
       do(reset weapon-selected) and
       do(reset ammunition-selected) and
       do(reset maybe-should-fire) and
       do(reset dismounted-check) and
       do(reset location-check) and
       do(reset formation-check) and
       do(reset unknown-near-road) and
       do(reset unknown-location) and
       do(reset unknown-multiple-target) and
       do(reset output-range) and
       do(reset output-weapon) and
       do(reset unknown-in-formation) and
       do(reset the-user-is-finished) and
       do(restart)
    then all-targets-engaged.

kb-48:
    if patrol-status and
       display('There is a possibility that the target is a friendly patrol.\n')

    then too-low-patrol.

kb-49:
    if not patrol-status
    then too-low-patrol.

kb-50:
    if civilian-status and
       display('There is a possibility that the target is a party of noncombatants.\n')

    then too-low-civilian.

kb-51:
    if not civilian-status
    then too-low-civilian.
```

```
kb-52:
    if mostlikely(the-range) = Range and
       display('\f \n \n \n \t The range to the target is ') and
       display(Range) and
       display(' meters.\n \n \n')
    then output-range.

kb-53:
    if mostlikely(weapon-selected) = Weapon and
       Weapon == indirect and
       mostlikely(ammunition-selected) = Ammunition and
       display('\t The preferred weapon is artillery fire because \n') and
       display('\t of the long distance to the target.\n  \n') and
       display('\t Request indirect fire through your fire support channels.')
        and
       display('\n \t The preferred munition is ') and
       display(Ammunition)
    then output-weapon.

kb-54:
    if mostlikely(weapon-selected) = Weapon and
       mostlikely(ammunition-selected) = Ammunition and
       display('\t The preferred weapon is the ') and
       display(Weapon) and
       display('\n \t firing ') and
       display(Ammunition) and
       display(' ammunition.')
    then output-weapon.

kb-55:
    explanation(self-defense) =
    ['\n Firing in self defense is always allowed.  Answering\n',
    'yes to this question will take the user to the targeting menus\n',
    'immediately.\n'].

kb-56:
    question(self-defense) =
    ['\f \n \n \t \t Is the target currently firing at you? \n \n'].

kb-57:
    legalvals(self-defense) = [yes,no].

kb-58:
    automaticmenu(self-defense).

kb-59:
    enumeratedanswers(self-defense).

kb-60:
    question(the-user-is-finished) =
```

```
    ['\n \t This concludes one iteration of the Bradley Gunner.  Do\n ',
    '\t you wish to continue?\n '].

kb-61:
    legalvals(the-user-is-finished) = [yes,no].

kb-62:
    automaticmenu(the-user-is-finished).

kb-63:
    enumeratedanswers(the-user-is-finished).

kb-64:
    if the-side = enemy
    then should-fire = yes.

kb-65:
    if the-side is sought and
       the-side = friendly
    then should-fire = no.

kb-66:
    if the-side is sought and
       the-side = not_known and
       status-check = SC and
       do(set maybe-should-fire = yes cf SC) and
       dismounted-check = DC and
       do(set maybe-should-fire = yes cf DC) and
       location-check = LC and
       do(set maybe-should-fire = yes cf LC) and
       formation-check = FC and
       do(set maybe-should-fire = yes cf FC) and
       maybe-should-fire = Final
    then should-fire = Final.

kb-67:
    if friendly-status = FS and
       enemy-status = ES and
       status-check-function(FS,ES) = SCF
    then status-check = SCF.

kb-68:
    status-check-function(leading,attacking) = 60.

kb-69:
    status-check-function(attacking,attacking) = 55.

kb-70:
    status-check-function(defending,attacking) = 60.
```

```
kb-71:
    status-check-function(in_reserve,attacking) = 50.

kb-72:
    status-check-function(reconnaissance/observation,attacking) = 10.

kb-73:
    status-check-function(withdrawing,attacking) = 50.

kb-74:
    status-check-function(delay,attacking) = 50.

kb-75:
    status-check-function(leading,defending) = 55.

kb-76:
    status-check-function(attacking,defending) = 50.

kb-77:
    status-check-function(defending,defending) = 40.

kb-78:
    status-check-function(in_reserve,defending) = 30.

kb-79:
    status-check-function(reconnaissance/observation,defending) = 5.

kb-80:
    status-check-function(withdrawing,defending) = 5.

kb-81:
    status-check-function(delay,defending) = 5.

kb-82:
    status-check-function(leading,withdrawing) = 60.

kb-83:
    status-check-function(attacking,withdrawing) = 55.

kb-84:
    status-check-function(defending,withdrawing) = 30.

kb-85:
    status-check-function(in_reserve,withdrawing) = 20.

kb-86:
    status-check-function(reconnaissance/observation,withdrawing) = 10.

kb-87:
    status-check-function(withdrawing,withdrawing) = 5.
```

```
kb-88:
    status-check-function(delay,withdrawing) = 5.

kb-89:
    status-check-function(leading,delay) = 60.

kb-90:
    status-check-function(attacking,delay) = 55.

kb-91:
    status-check-function(defending,delay) = 30.

kb-92:
    status-check-function(in_reserve,delay) = 20.

kb-93:
    status-check-function(reconnaissance/observation,delay) = 10.

kb-94:
    status-check-function(withdrawing,delay) = 5.

kb-95:
    status-check-function(delay,delay) = 5.

kb-96:
    if the-target = UTT and
       patrol-status = PS and
       civilian-status = CS and
       dismounted-check-function(UTT,PS,CS) = DCF
    then dismounted-check = DCF.

kb-97:
    dismounted-check-function(infantry,no,no) = 10.

kb-98:
    dismounted-check-function(infantry,no,yes) = -10.

kb-99:
    dismounted-check-function(infantry,yes,no) = -20.

kb-100:
    dismounted-check-function(infantry,yes,yes) = -30.

kb-101:
    dismounted-check-function(dismounted,no,no) = 10.

kb-102:
    dismounted-check-function(dismounted,no,yes) = -10.
```

```
kb-103:
    dismounted-check-function(dismounted,yes,no) = -20.

kb-104:
    dismounted-check-function(dismounted,yes,yes) = -30.

kb-105:
    dismounted-check-function(uncertain,no,no) = 10.

kb-106:
    dismounted-check-function(uncertain,no,yes) = -10.

kb-107:
    dismounted-check-function(uncertain,yes,no) = -20.

kb-108:
    dismounted-check-function(uncertain,yes,yes) = -30.

kb-109:
    dismounted-check-function(ANY,no,no) = 40.

kb-110:
    dismounted-check-function(ANY,no,yes) = 30.

kb-111:
    dismounted-check-function(ANY,yes,no) = 20.

kb-112:
    dismounted-check-function(ANY,yes,yes) = 10.

kb-113:
    explanation(the-target) =
    ['Different types of targets require different type of weapons and \n',
    'ammunitions to destroy them.  Answering this question will let the \n',
    'gunner choose the best weapon and ammunition combination.'].

kb-114:
    question(the-target) = '\f \n \t What type of target is it? \n '.

kb-115:
    legalvals(the-target) = [tank,apc,sp_artillery,towed_artillery,
    anti_tank_unit,infantry,trucks,logistics_area,command_post,tracked,
    wheeled,dismounted,uncertain].

kb-116:
    automaticmenu(the-target).

kb-117:
    enumeratedanswers(the-target).
```

```
kb-118:
    if unknown-near-road = UNR and
       unknown-location = UL and
       location-check-function(UNR,UL) = LCF
    then location-check = LCF.

kb-119:
    explanation(unknown-near-road) =
    ['You have stated that the identity of the target is not known. \n',
    'This question will assist the gunner in determining whether \n',
    'or not to fire at the target.'].

kb-120:
    question(unknown-near-road) =
    '\f \n \t Is the unknown target on, near or away from a road?\n '.

kb-121:
    legalvals(unknown-near-road) = [on,near,away].

kb-122:
    automaticmenu(unknown-near-road).

kb-123:
    enumeratedanswers(unknown-near-road).

kb-124:
    explanation(unknown-location) =
    ['You have stated that the identity of the target is not known. \n',
    'This question will assist the gunner in determining whether \n',
    'or not to fire at the target.'].
```

```
kb-125:
    question(unknown-location) =
    '\n \t Is the target near a known enemy or friendly location? \n '.

kb-126:
    legalvals(unknown-location) = [enemy,friendly,neither].

kb-127:
    automaticmenu(unknown-location).

kb-128:
    enumeratedanswers(unknown-location).

kb-129:
    location-check-function(on,enemy) = 20.

kb-130:
    location-check-function(near,enemy) = 25.

kb-131:
    location-check-function(away,enemy) = 30.

kb-132:
    location-check-function(on,friendly) = -20.

kb-133:
    location-check-function(near,friendly) = -20.

kb-134:
    location-check-function(away,friendly) = -20.

kb-135:
    location-check-function(on,neither) = 0.

kb-136:
    location-check-function(near,neither) = 0.

kb-137:
    location-check-function(away,neither) = 10.

kb-138:
    if unknown-multiple-target = UMT and
       unknown-in-formation = UF and
       formation-check-function(UMT,UF) = FCF
    then formation-check = FCF.
```

```
kb-139:
    explanation(unknown-multiple-target) =
    ['You have stated that the identity of the target is not known. \n',
    'This question will assist the gunner in determining whether \n',
    'or not to fire at the target.'].

kb-140:
    question(unknown-multiple-target) =
    '\f \n Are there more than one unknown targets? \n '.

kb-141:
    legalvals(unknown-multiple-target) = [yes,no].

kb-142:
    automaticmenu(unknown-multiple-target).

kb-143:
    enumeratedanswers(unknown-multiple-target).

kb-144:
    explanation(unknown-in-formation) =
    ['You have stated that the identity of the target is not known. \n',
    'This question will assist the gunner in determining whether \n',
    'or not to fire at the target.'].

kb-145:
    question(unknown-in-formation) = '\nIs the target(s) in a formation?\n '.

kb-146:
    legalvals(unknown-in-formation) = [yes,no].

kb-147:
    automaticmenu(unknown-in-formation).

kb-148:
    enumeratedanswers(unknown-in-formation).

kb-149:
    formation-check-function(no,no) = 0.

kb-150:
    formation-check-function(no,yes) = 0.

kb-151:
    formation-check-function(yes,no) = 10.
```

```
kb-152:
    formation-check-function(yes,yes) = 15.

kb-153:
    explanation(the-side) = ['Some type of determination must be made. \n ',
    'You must answer this question to continue.\n'].

kb-154:
    question(the-side) =
    '\n \tIs the target enemy, friendly, or not known?\n'.

kb-155:
    legalvals(the-side) = [enemy,friendly,not_known].

kb-156:
    automaticmenu(the-side).

kb-157:
    enumeratedanswers(the-side).

kb-158:
    if (the-target = tank or
        (the-target = uncertain and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = infantry)) and
       the-range = Range and
       Range>= 3751 and
       weapon-type(tank,extreme) = Weapon_return
    then weapon-selected = Weapon_return.

kb-159:
    if (the-target = tank or
        (the-target = uncertain and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = infantry)) and
       the-range = Range and
       Range<3751 and
       Range>= 65 and
       weapon-type(tank,long) = Weapon_return
    then weapon-selected = Weapon_return.

kb-160:
    if (the-target = tank or
        (the-target = uncertain and
```

```
        enemy-unit-type = tank) or
       (the-target = tracked and
        enemy-unit-type = tank) or
       (the-target = tracked and
        enemy-unit-type = infantry)) and
      the-range = Range and
      Range<65 and
      weapon-type(tank,close) = Weapon_return
   then weapon-selected = Weapon_return.

kb-161:
   if (the-target = apc or
       (the-target = uncertain and
        enemy-unit-type = apc) or
       (the-target = tracked and
        enemy-unit-type = apc)) and
      the-range = Range and
      Range>= 3751 and
      weapon-type(apc,extreme) = Weapon_return
   then weapon-selected = Weapon_return.

kb-162:
   if (the-target = apc or
       (the-target = uncertain and
        enemy-unit-type = apc) or
       (the-target = tracked and
        enemy-unit-type = apc)) and
      the-range = Range and
      Range<3751 and
      Range>= 1700 and
      weapon-type(apc,long) = Weapon_return
   then weapon-selected = Weapon_return.

kb-163:
   if (the-target = apc or
       (the-target = uncertain and
        enemy-unit-type = apc) or
       (the-target = tracked and
        enemy-unit-type = apc)) and
      the-range = Range and
      Range<1700 and
      weapon-type(apc,medium) = Weapon_return
   then weapon-selected = Weapon_return.

kb-164:
   if the-target = sp_artillery and
      the-range = Range and
      Range>= 3751 and
      weapon-type(apc,extreme) = Weapon_return
   then weapon-selected = Weapon_return.
```

```
kb-165:
    if the-target = sp_artillery and
       the-range = Range and
       Range<3751 and
       Range>1700 and
       weapon-type(apc,long) = Weapon_return
    then weapon-selected = Weapon_return.

kb-166:
    if the-target = sp_artillery and
       the-range = Range and
       Range<= 1700 and
       weapon-type(apc,medium) = Weapon_return
    then weapon-selected = Weapon_return.

kb-167:
    if the-target = towed_artillery and
       the-range = Range and
       Range>3000 and
       weapon-type(equipment,long) = Weapon_return
    then weapon-selected = Weapon_return.

kb-168:
    if the-target = towed_artillery and
       the-range = Range and
       Range<= 3000 and
       weapon-type(equipment,medium) = Weapon_return
    then weapon-selected = Weapon_return.

kb-169:
    if the-target = anti_tank_unit and
       the-range = Range and
       Range>3000 and
       weapon-type(equipment,long) = Weapon_return
    then weapon-selected = Weapon_return.
```

```
kb-170:
    if the-target = anti_tank_unit and
       the-range = Range and
       Range<= 3000 and
       weapon-type(equipment,medium) = Weapon_return
    then weapon-selected = Weapon_return.

kb-171:
    if (the-target = infantry or
        (the-target = uncertain and
         enemy-unit-type = infantry) or
        (the-target = dismounted and
         enemy-unit-type = infantry)) and
       the-range = Range and
       Range>= 3000 and
       weapon-type(infantry,long) = Weapon_return
    then weapon-selected = Weapon_return.

kb-172:
    if (the-target = infantry or
        (the-target = uncertain and
         enemy-unit-type = infantry) or
        (the-target = dismounted and
         enemy-unit-type = infantry)) and
       the-range = Range and
       Range<3000 and
       Range>= 800 and
       weapon-type(infantry,medium) = Weapon_return
    then weapon-selected = Weapon_return.

kb-173:
    if (the-target = infantry or
        (the-target = uncertain and
         enemy-unit-type = infantry) or
        (the-target = dismounted and
         enemy-unit-type = infantry)) and
       the-range = Range and
       Range<= 800 and
       weapon-type(infantry,close) = Weapon_return
    then weapon-selected = Weapon_return.
```

```
kb-174:
    if (the-target = trucks or
        the-target = wheeled) and
       the-range = Range and
       Range>= 3000 and
       weapon-type(infantry,long) = Weapon_return
    then weapon-selected = Weapon_return.

kb-175:
    if (the-target = trucks or
        the-target = wheeled) and
       the-range = Range and
       Range<3000 and
       Range>= 800 and
       weapon-type(infantry,medium) = Weapon_return
    then weapon-selected = Weapon_return.

kb-176:
    if (the-target = trucks or
        the-target = wheeled) and
       the-range = Range and
       Range<= 800 and
       weapon-type(infantry,close) = Weapon_return
    then weapon-selected = Weapon_return.

kb-177:
    if the-target = logistics_area and
       the-range = Range and
       Range>= 3000 and
       weapon-type(cp,long) = Weapon_return
    then weapon-selected = Weapon_return.

kb-178:
    if the-target = logistics_area and
       the-range = Range and
       Range<3000 and
       Range>100 and
       weapon-type(cp,medium) = Weapon_return
    then weapon-selected = Weapon_return.

kb-179:
    if the-target = logistics_area and
       the-range = Range and
       Range<= 100 and
       weapon-type(cp,close) = Weapon_return
    then weapon-selected = Weapon_return.

kb-180:
    if the-target = command_post and
       the-range = Range and
```

```
      Range>= 3000 and
      weapon-type(cp,long) = Weapon_return
   then weapon-selected = Weapon_return.

kb-181:
   if the-target = command_post and
      the-range = Range and
      Range<3000 and
      Range>100 and
      weapon-type(cp,medium) = Weapon_return
   then weapon-selected = Weapon_return.

kb-182:
   if the-target = command_post and
      the-range = Range and
      Range<= 100 and
      weapon-type(cp,close) = Weapon_return
   then weapon-selected = Weapon_return.

kb-183:
   explanation(the-range) =
   ['This is required data.  Guess if necessary.\n'].

kb-184:
   question(the-range) = '\n \t What is the range to the target in meters?'.

kb-185:
   legalvals(the-range) = number.

kb-186:
   explanation(the-terrain) =
   ['Indirect fire has been chosen as the best weapon.\n',
   'What the terrain is like will influence the preferred shell and\n',
   'shell and fuze combination.\n'].

kb-187:
   question(the-terrain) =
   '\n \t What type of terrain is the target in? \n '.

kb-188:
   legalvals(the-terrain) = [open,wooded,trenches/dug_in,town].

kb-189:
   automaticmenu(the-terrain).

kb-190:
   enumeratedanswers(the-terrain).

kb-191:
   weapon-type(tank,extreme) = indirect.
```

```
kb-192:
    weapon-type(tank,long) = guided_missile.

kb-193:
    weapon-type(tank,medium) = guided_missile.

kb-194:
    weapon-type(tank,close) = automatic_cannon.

kb-195:
    weapon-type(apc,extreme) = indirect.

kb-196:
    weapon-type(apc,long) = guided_missile.

kb-197:
    weapon-type(apc,medium) = automatic_cannon.

kb-198:
    weapon-type(apc,close) = automatic_cannon.

kb-199:
    weapon-type(equipment,extreme) = indirect.

kb-200:
    weapon-type(equipment,long) = indirect.

kb-201:
    weapon-type(equipment,medium) = automatic_cannon.

kb-202:
    weapon-type(equipment,close) = automatic_cannon.

kb-203:
    weapon-type(infantry,extreme) = indirect.

kb-204:
    weapon-type(infantry,long) = indirect.
```

```
kb-205:
    weapon-type(infantry,medium) = automatic_cannon.

kb-206:
    weapon-type(infantry,close) = machinegun.

kb-207:
    weapon-type(cp,extreme) = indirect.

kb-208:
    weapon-type(cp,long) = indirect.

kb-209:
    weapon-type(cp,medium) = automatic_cannon.

kb-210:
    weapon-type(cp,close) = machinegun.

kb-211:
    if weapon-selected = guided_missile
    then ammunition-selected = anti_tank.

kb-212:
    if weapon-selected = automatic_cannon and
       the-target = uncertain and
       enemy-unit-type = R and
       ammunition-type(automatic_cannon,R) = Ammunition_return
    then ammunition-selected = Ammunition_return.

kb-213:
    if weapon-selected = automatic_cannon and
       the-target = R and
       ammunition-type(automatic_cannon,R) = Ammunition_return
    then ammunition-selected = Ammunition_return.

kb-214:
    if weapon-selected = machinegun
    then ammunition-selected = ball/tracer.

kb-215:
    if weapon-selected = indirect and
       the-terrain is sought and
       the-terrain = T and
       the-target = uncertain and
       enemy-unit-type = R and
       artillery-type(T,R) = Artillery_return
    then ammunition-selected = Artillery_return.

kb-216:
    if weapon-selected = indirect and
```

```
        the-terrain is sought and
        the-terrain = T and
        the-target = R and
        artillery-type(T,R) = Artillery_return
    then ammunition-selected = Artillery_return.

kb-217:
    ammunition-type(automatic_cannon,tank) = armor_piercing_discarding_sabot.

kb-218:
    ammunition-type(automatic_cannon,apc) = armor_piercing_discarding_sabot.

kb-219:
    ammunition-type(automatic_cannon,sp_artillery) =
    armor_piercing_discarding_sabot.

kb-220:
    ammunition-type(automatic_cannon,towed_artillery) =
    high_explosive_incendiary.

kb-221:
    ammunition-type(automatic_cannon,anti_tank_unit) =
    high_explosive_incendiary.

kb-222:
    ammunition-type(automatic_cannon,infantry) = high_explosive_incendiary.

kb-223:
    ammunition-type(automatic_cannon,trucks) = high_explosive_incendiary.

kb-224:
    ammunition-type(automatic_cannon,logistics_area) =
    high_explosive_incendiary.

kb-225:
    ammunition-type(automatic_cannon,command_post) =
    high_explosive_incendiary.

kb-226:
    ammunition-type(automatic_cannon,uncertain) = high_explosive_incendiary.

kb-227:
    ammunition-type(automatic_cannon,tracked) =
    armor_piercing_discarding_sabot.

kb-228:
    ammunition-type(automatic_cannon,wheeled) = high_explosive_incendiary.

kb-229:
    ammunition-type(automatic_cannon,dismounted) = high_explosive_incendiary.
```

```
kb-230:
    ammunition-type(automatic_cannon,uncertain) = high_explosive_incendiary.

kb-231:
    artillery-type(open,tank) = dual_purpose_improved_conventional_munition.

kb-232:
    artillery-type(open,apc) = dual_purpose_improved_conventional_munition.

kb-233:
    artillery-type(open,sp_artillery) =
    dual_purpose_improved_conventional_munition.

kb-234:
    artillery-type(open,towed_artillery) = high_explosive_vt.

kb-235:
    artillery-type(open,anti_tank_unit) = high_explosive_vt.

kb-236:
    artillery-type(open,infantry) = high_explosive_vt.

kb-237:
    artillery-type(open,trucks) = high_explosive_vt.

kb-238:
    artillery-type(open,logistics_area) = high_explosive_and_white_phosphorus.

kb-239:
    artillery-type(open,command_post) = high_explosive_and_white_phosphorus.

kb-240:
    artillery-type(open,tracked) =
    dual_purpose_improved_conventional_munition.

kb-241:
    artillery-type(open,wheeled) = high_explosive_vt.

kb-242:
    artillery-type(open,dismounted) = high_explosive_vt.

kb-243:
    artillery-type(open,uncertain) = high_explosive.

kb-244:
    artillery-type(wooded,tank) = high_explosive_delay.

kb-245:
    artillery-type(wooded,apc) = high_explosive_delay.
```

```
kb-246:
    artillery-type(wooded,sp_artillery) = high_explosive_delay.

kb-247:
    artillery-type(wooded,towed_artillery) = high_explosive_delay.

kb-248:
    artillery-type(wooded,anti_tank_unit) = high_explosive_delay.

kb-249:
    artillery-type(wooded,infantry) = high_explosive_delay.

kb-250:
    artillery-type(wooded,trucks) = high_explosive_delay.

kb-251:
    artillery-type(wooded,logistics_area) =
    high_explosive_delay_and_white_phosphorus.

kb-252:
    artillery-type(wooded,command_post) =
    high_explosive_delay_and_white_phosphorus.

kb-253:
    artillery-type(wooded,tracked) = high_explosive_delay.

kb-254:
    artillery-type(wooded,wheeled) = high_explosive_delay.

kb-255:
    artillery-type(wooded,dismounted) = high_explosive_delay.

kb-256:
    artillery-type(wooded,uncertain) = high_explosive_delay.
```

```
kb-257:
    artillery-type(trenches/dug_in,tank) =
    dual_purpose_improved_conventional_munition.

kb-258:
    artillery-type(trenches/dug_in,apc) =
    dual_purpose_improved_conventional_munition.

kb-259:
    artillery-type(trenches/dug_in,sp_artillery) =
    dual_purpose_improved_conventional_munition.

kb-260:
    artillery-type(trenches/dug_in,towed_artillery) =
    high_explosive_vt_and_high_explosive_delay.

kb-261:
    artillery-type(trenches/dug_in,anti_tank_unit) =
    high_explosive_vt_and_high_explosive_delay.

kb-262:
    artillery-type(trenches/dug_in,infantry) =
    high_explosive_vt_and_high_explosive_delay.

kb-263:
    artillery-type(trenches/dug_in,trucks) = high_explosive_vt.

kb-264:
    artillery-type(trenches/dug_in,logistics_area) =
    high_explosive_vt_and_high_explosive_delay.

kb-265:
    artillery-type(trenches/dug_in,command_post) =
    high_explosive_vt_and_high_explosive_delay.

kb-266:
    artillery-type(trenches/dug_in,tracked) =
    dual_purpose_improved_conventional_munition.

kb-267:
    artillery-type(trenches/dug_in,wheeled) =
    high_explosive_vt_and_high_explosive_delay.

kb-268:
    artillery-type(trenches/dug_in,dismounted) =
    high_explosive_vt_and_high_explosive_delay.

kb-269:
    artillery-type(trenches/dug_in,uncertain) =
    high_explosive_vt_and_high_explosive_delay.
```

```
kb-270:
    artillery-type(town,tank) =
    dual_purpose_improved_conventional_munition_and_high_explosive_delay.

kb-271:
    artillery-type(town,apc) =
    dual_purpose_improved_conventional_munition_and_high_explosive_delay.

kb-272:
    artillery-type(town,sp_artillery) =
    dual_purpose_improved_conventional_munition_and_high_explosive_delay.

kb-273:
    artillery-type(town,towed_artillery) =
    high_explosive_and_high_explosive_delay.

kb-274:
    artillery-type(town,anti_tank_unit) =
    high_explosive_and_high_explosive_delay.

kb-275:
    artillery-type(town,infantry) = high_explosive_and_high_explosive_delay.

kb-276:
    artillery-type(town,trucks) = high_explosive_and_high_explosive_delay.

kb-277:
    artillery-type(town,logistics_area) = high_explosive_and_white_phosphorus.

kb-278:
    artillery-type(town,command_post) =
    high_explosive_and_high_explosive_delay.

kb-279:
    artillery-type(town,tracked) =
    dual_purpose_improved_conventional_munition_and_high_explosive_delay.

kb-280:
    artillery-type(town,wheeled) = high_explosive_and_high_explosive_delay.

kb-281:
    artillery-type(town,dismounted) = high_explosive_and_high_explosive_delay.

kb-282:
    artillery-type(town,uncertain) = high_explosive_and_high_explosive_delay.
```

APPENDIX III

This appendix contains a photo session of the expert gunnery system. This particular session is **without** a rule base trace. The expert system's questions and the user's responses appear here as they do on the screen.

```
M.1> go

        The M2/M3 Bradley Fighting Vehicle Gunner Advisor

        A system to assist a vehicle commander in deciding
        whether or not to fire on known or unknown targets,
        and to provide the commander with the correct weapon
        and ammunition combination for the job.

           Built by CPT Timothy J. Gibson.

 In order to run this weapon adviser the user needs to know
 or have access to the range to specific targets or a rangefinder,
 a map, and current intelligence reports.


           Are you prepared at this time?
    1. yes
    2. no
>> 1

      Background Questions

     What is your units' current tactical posture?

    1. attacking
    2. defending
    3. in_reserve
    4. reconnaissance/observation
    5. withdrawing
    6. delay
>> 1

      Is your unit leading the attack?

    1. yes
    2. no
>> 1

      Background Questions
```

What is enemy's current tactical posture?

1. attacking
2. defending
3. withdrawing
4. delay

>> 2

What type of enemy unit are you currently facing

1. tank
2. apc
3. infantry

>> 1

Background Questions

Are there friendly patrols operating in this area?

1. yes
2. no

>> 2

Are there civilian refugees in this area?

1. yes
2. no

>> 1

Is the target currently firing at you?

1. yes
2. no

>> 1

Firing in self defense is always authorized.

```
     What type of target is it?

   1. tank
   2. apc
   3. sp_artillery
   4. towed_artillery
   5. anti_tank_unit
   6. infantry
   7. trucks
   8. logistics_area
   9. command_post
   10. tracked
   11. wheeled
   12. dismounted
   13. uncertain
>> 2

     What is the range to the target in meters?
>> 1500


     The range to the target is 1500 meters.


     The preferred weapon is the automatic_cannon
     firing armor_piercing_discarding_sabot ammunition.

     This concludes one iteration of the Bradley Gunner.  Do
     you wish to continue?

   1. yes
   2. no
>> 1


          Is the target currently firing at you?


   1. yes
   2. no
>> 2
```

```
     Is the target enemy, friendly, or not known?

    1.  enemy
    2.  friendly
    3.  not_known
>> 2


      Do not shoot at friendly troops.


      This concludes one iteration of the Bradley Gunner.  Do
      you wish to continue?

    1.  yes
    2.  no
>> 1


           Is the target currently firing at you?


    1.  yes
    2.  no
>> 2

     Is the target enemy, friendly, or not known?

    1.  enemy
    2.  friendly
    3.  not_known
>> 1
```

```
     What type of target is it?

    1.  tank
    2.  apc
    3.  sp_artillery
    4.  towed_artillery
    5.  anti_tank_unit
    6.  infantry
    7.  trucks
    8.  logistics_area
    9.  command_post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 4

     What is the range to the target in meters?
>> 3500

     What type of terrain is the target in?

    1.  open
    2.  wooded
    3.  trenches/dug_in
    4.  town
>> 2


     The range to the target is 3500 meters.


     The preferred weapon is artillery fire because
     of the long distance to the target.

     Request indirect fire through your fire support channels.
     The preferred munition is high_explosive_delay

     This concludes one iteration of the Bradley Gunner.  Do
     you wish to continue?

    1.  yes
    2.  no
>> 1


          Is the target currently firing at you?
```

```
    1. yes
    2. no
>> 2

     Is the target enemy, friendly, or not known?

    1. enemy
    2. friendly
    3. not_known
>> 3

      What type of target is it?

    1. tank
    2. apc
    3. sp_artillery
    4. towed_artillery
    5. anti_tank_unit
    6. infantry
    7. trucks
    8. logistics_area
    9. command_post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 10

      Is the unknown target on, near or away from a road?

    1. on
    2. near
    3. away
>> 3

      Is the target near a known enemy or friendly location?

    1. enemy
    2. friendly
    3. neither
>> 1
```

```
 Are there more than one unknown targets?

    1. yes
    2. no
>> 1

Is the target(s) in a formation?

    1. yes
    2. no
>> 1

      What is the range to the target in meters?
>> 2500



      The range to the target is 2500 meters.


      The preferred weapon is the guided_missile
      firing anti_tank ammunition.

      This concludes one iteration of the Bradley Gunner.  Do
      you wish to continue?

    1. yes
    2. no
>> 2
```

This next section contains another iteration of the Bradley Gunner with different background data.

```
M.1> go

         The M2/M3 Bradley Fighting Vehicle Gunner Advisor

         A system to assist a vehicle commander in deciding
         whether or not to fire on known or unknown targets,
         and to provide the commander with the correct weapon
         and ammunition combination for the job.

            Built by CPT Timothy J. Gibson.
```

```
In order to run this weapon adviser the user needs to know
or have access to the range to specific targets or a rangefinder,
a map, and current intelligence reports.


          Are you prepared at this time?
   1. yes
   2. no
>> 1

      Background Questions

     What is your units' current tactical posture?

   1. attacking
   2. defending
   3. in_reserve
   4. reconnaissance/observation
   5. withdrawing
   6. delay
>> 1

      Is your unit leading the attack?

   1. yes
   2. no
>> 1

      Background Questions

     What is enemy's current tactical posture?

   1. attacking
   2. defending
   3. withdrawing
   4. delay
>> 3

      What type of enemy unit are you currently facing

   1. tank
   2. apc
   3. infantry
>> 2
```

```
     Background Questions

     Are there friendly patrols operating in this area?

    1.  yes
    2.  no
>> 2

      Are there civilian refugees in this area?

    1.  yes
    2.  no
>> 2


          Is the target currently firing at you?


    1.  yes
    2.  no
>> 1

      Firing in self defense is always authorized.

      What type of target is it?

    1.  tank
    2.  apc
    3.  sp_artillery
    4.  towed_artillery
    5.  anti_tank_unit
    6.  infantry
    7.  trucks
    8.  logistics_area
    9.  command_post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 4

      What is the range to the target in meters?
>> 3200
```

```
      What type of terrain is the target in?

    1. open
    2. wooded
    3. trenches/dug_in
    4. town
>> 2


      The range to the target is 3200 meters.


      The preferred weapon is artillery fire because
      of the long distance to the target.

      Request indirect fire through your fire support channels.
      The preferred munition is high_explosive_delay

      This concludes one iteration of the Bradley Gunner.  Do
      you wish to continue?

    1. yes
    2. no
>> 1


           Is the target currently firing at you?


    1. yes
    2. no
>> 2

     Is the target enemy, friendly, or not known?

    1. enemy
    2. friendly
    3. not_known
>> 1
```

```
     What type of target is it?

    1.  tank
    2.  apc
    3.  sp_artillery
    4.  towed_artillery
    5.  anti_tank_unit
    6.  infantry
    7.  trucks
    8.  logistics_area
    9.  command_post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 2

     What is the range to the target in meters?
>> 1500


     The range to the target is 1500 meters.


     The preferred weapon is the automatic_cannon
     firing armor_piercing_discarding_sabot ammunition.

     This concludes one iteration of the Bradley Gunner.  Do
     you wish to continue?

    1.  yes
    2.  no
>> 1


          Is the target currently firing at you?


    1.  yes
    2.  no
>> 2
```

```
     Is the target enemy, friendly, or not known?

    1.  enemy
    2.  friendly
    3.  not_known
>> 1

      What type of target is it?

    1.  tank
    2.  apc
    3.  sp_artillery
    4.  towed_artillery
    5.  anti_tank_unit
    6.  infantry
    7.  trucks
    8.  logistics_area
    9.  command_post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 2

      What is the range to the target in meters?
>> 2500


      The range to the target is 2500 meters.


      The preferred weapon is the guided_missile
      firing anti_tank ammunition.

      This concludes one iteration of the Bradley Gunner.  Do
      you wish to continue?

    1.  yes
    2.  no
>> 1
```

```
          Is the target currently firing at you?

    1.  yes
    2.  no
>> 2

     Is the target enemy, friendly, or not known?

    1.  enemy
    2.  friendly
    3.  not_known
>> 1

      What type of target is it?

    1.  tank
    2.  apc
    3.  sp_artillery
    4.  towed_artillery
    5.  anti_tank_unit
    6.  infantry
    7.  trucks
    8.  logistics_area
    9.  command_post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 8

      What is the range to the target in meters?
>> 3200

      What type of terrain is the target in?

    1.  open
    2.  wooded
    3.  trenches/dug_in
    4.  town
>> 2
```

```
      The range to the target is 3200 meters.


      The preferred weapon is artillery fire because
      of the long distance to the target.

      Request indirect fire through your fire support channels.
      The preferred munition is high_explosive_delay_and_white_phosphorus

      This concludes one iteration of the Bradley Gunner.  Do
      you wish to continue?

    1.  yes
    2.  no
>> 1



          Is the target currently firing at you?


    1.  yes
    2.  no
>> 2

     Is the target enemy, friendly, or not known?

    1.  enemy
    2.  friendly
    3.  not_known
>> 3
```

```
     What type of target is it?

    1.  tank
    2.  apc
    3.  sp_artillery
    4.  towed_artillery
    5.  anti_tank_unit
    6.  infantry
    7.  trucks
    8.  logistics_area
    9.  command_post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 10

     Is the unknown target on, near or away from a road?

    1.  on
    2.  near
    3.  away
>> 3

     Is the target near a known enemy or friendly location?

    1.  enemy
    2.  friendly
    3.  neither
>> 1

 Are there more than one unknown targets?

    1.  yes
    2.  no
>> 1

Is the target(s) in a formation?

    1.  yes
    2.  no
>> 1
```

```
     What is the range to the target in meters?
>> 1750


     The range to the target is 1750 meters.


     The preferred weapon is the guided_missile
     firing anti_tank ammunition.

     This concludes one iteration of the Bradley Gunner.  Do
     you wish to continue?

   1. yes
   2. no
>> 1


          Is the target currently firing at you?


   1. yes
   2. no
>> 2

     Is the target enemy, friendly, or not known?

   1. enemy
   2. friendly
   3. not_known
>> 1
```

```
      What type of target is it?

    1.  tank
    2.  apc
    3.  sp_artillery
    4.  towed_artillery
    5.  anti_tank_unit
    6.  infantry
    7.  trucks
    8.  logistics_area
    9.  command_post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 6

      What is the range to the target in meters?
>> 500


      The range to the target is 500 meters.


      The preferred weapon is the machinegun
      firing ball/tracer ammunition.

      This concludes one iteration of the Bradley Gunner.  Do
      you wish to continue?

    1.  yes
    2.  no
>> 2

M.1> logoff
```

APPENDIX IV

This appendix contains a photo session with the expert gunnery system. This session is **with** a rule base trace. The expert system's questions and the user's responses appear are shown with an underline to identify them.

M.1> go

```
Seeking begin the-consultation.
Invoking kb-8:
    if begin message = M and
       display(M) and
       begin signal and
       display('\f') and
       friendly-status is sought and
       enemy-status is sought and
       enemy-unit-type is sought and
       patrol-status is sought and
       civilian-status is sought and
       display('\f')
    then begin the-consultation.
Seeking begin message.
Using kb-2:
    begin message = [nl,' ',nl,
    '         The M2/M3 Bradley Fighting Vehicle Gunner Advisor ',nl,' ',nl,
    '         A system to assist a vehicle commander in deciding ',nl,
    '         whether or not to fire on known or unknown targets,',nl,
    '         and to provide the commander with the correct weapon',nl,
    '         and ammunition combination for the job.              ',nl,
    '                                                              ',nl,
    '            Built by CPT Timothy J. Gibson.'].
Noting begin message = [nl,' ',nl,'         The M2/M3 Bradley Fighting Vehicle Gunner Advisor
',nl,' ',nl,'         A system to assist a vehicle commander in deciding ',nl,'
whether or not to fire on known or unknown targets,',nl,'         and to provide the
commander with the correct weapon',nl,'         and ammunition combination for the job.
',nl,'                                                              ',nl,'            Built
by CPT Timothy J. Gibson.'] cf 100 because kb-2.
Found begin message.
```

```
        The M2/M3 Bradley Fighting Vehicle Gunner Advisor
        A system to assist a vehicle commander in deciding
        whether or not to fire on known or unknown targets,
        and to provide the commander with the correct weapon
        and ammunition combination for the job.

             Built by CPT Timothy J. Gibson.
Seeking begin signal.
Using kb-4:
    question(begin signal) = [nl,
    ' In order to run this weapon adviser the user needs to know',nl,
    ' or have access to the range to specific targets or a rangefinder,',nl,
    ' a map, and current intelligence reports.',nl,'  ',nl,'  ',nl,
    '          Are you prepared at this time?'].

 In order to run this weapon adviser the user needs to know
 or have access to the range to specific targets or a rangefinder,
 a map, and current intelligence reports.


             Are you prepared at this time?
    1. yes
    2. no
>> 1
Noting begin signal = yes cf 100 because you said so.
Found begin signal.
Seeking friendly-status.
Invoking kb-20:
    if friendly-determination = attacking and
       friendly-leading = yes
    then friendly-status = leading.
Seeking friendly-determination.
Using kb-11:
    question(friendly-determination) =
    ['\n \t Background Questions \n\n\tWhat is your units'' current tactical posture?\n']
    .
```

```
     Background Questions

    What is your units' current tactical posture?

    1. attacking
    2. defending
    3. in reserve
    4. reconnaissance/observation
    5. withdrawing
    6. delay
>> 2
Noting friendly-determination = defending cf 100 because you said so.
Found friendly-determination.
kb-20 failed.
Invoking kb-21:
    if friendly-determination = Anything_else
    then friendly-status = Anything_else.
Already sought friendly-determination.
Noting friendly-status = defending cf 100 because kb-21.
kb-21 succeeded.
Found friendly-status.
Seeking enemy-status.
Using kb-23:
    question(enemy-status) =
    ['\f \n\t Background Questions\n\n\tWhat is enemy''s current tactical posture?\n']
    .

     Background Questions

    What is enemy's current tactical posture?

    1. attacking
    2. defending
    3. withdrawing
    4. delay
>> 1
Noting enemy-status = attacking cf 100 because you said so.
Found enemy-status.
Seeking enemy-unit-type.
Using kb-28:
    question(enemy-unit-type) =
    ['\n \t What type of enemy unit are you currently facing \n'].
```

```
    What type of enemy unit are you currently facing

  1. tank
  2. apc
  3. infantry
>> 1
Noting enemy-unit-type = tank cf 100 because you said so.
Found enemy-unit-type.
Seeking patrol-status.
Using kb-33:
    question(patrol-status) = ['\f \n\tBackground Questions\n\n',
    '\tAre there friendly patrols operating in this area? \n'].

    Background Questions

    Are there friendly patrols operating in this area?

  1. yes
  2. no
>> 2
Noting patrol-status = no cf 100 because you said so.
Found patrol-status.
Seeking civilian-status.
Using kb-38:
    question(civilian-status) =
    ['\n \t Are there civilian refugees in this area? \n'].

    Are there civilian refugees in this area?

  1. yes
  2. no
>> 2
Noting civilian-status = no cf 100 because you said so.
Found civilian-status.
Noting begin the-consultation = yes cf 100 because kb-8.
kb-8 succeeded.
Found begin the-consultation.
Seeking continue the-consultation.
Invoking kb-42:
    if should-fire = no and
       display('\f \n \n \n \t Do not shoot at friendly troops.\n \n ') and
       the-user-is-finished = Answer
    then continue the-consultation = Answer.
Seeking should-fire.
```

```
Invoking kb-43:
    if self-defense = yes and
       display('\n \t Firing in self defense is always authorized. \n')
    then should-fire = yes cf 100.
Seeking self-defense.
Using kb-56:
    question(self-defense) =
    ['\f \n \n \t \t Is the target currently firing at you? \n \n'].


          Is the target currently firing at you?


    1. yes
    2. no
>> 1
Noting self-defense = yes cf 100 because you said so.
Found self-defense.

      Firing in self defense is always authorized.
Noting should-fire = yes cf 100 because kb-43.
kb-43 succeeded.
Found should-fire.
kb-42 failed.
Invoking kb-44:
    if self-defense is sought and
       should-fire is sought and
       should-fire cf N and
       N>= 80 and
       weapon-selected is sought and
       ammunition-selected is sought and
       output-range and
       output-weapon and
       the-user-is-finished = Answer
    then continue the-consultation = Answer.
Already sought self-defense.
Already sought should-fire.
Already sought should-fire.
Seeking weapon-selected.
```

```
Invoking kb-158:
    if (the-target = tank or
        (the-target = uncertain and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = infantry)) and
       the-range = Range and
       Range>= 3751 and
       weapon-type(tank,extreme) = Weapon_return
    then weapon-selected = Weapon_return.
Seeking the-target.
Using kb-114:
    question(the-target) = '\f \n \t What type of target is it? \n '.

      What type of target is it?

    1.  tank
    2.  apc
    3.  sp artillery
    4.  towed artillery
    5.  anti tank unit
    6.  infantry
    7.  trucks
    8.  logistics area
    9.  command post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 1
Noting the-target = tank cf 100 because you said so.
Found the-target.
Seeking the-range.
Using kb-184:
    question(the-range) = '\n \t What is the range to the target in meters?'.
```

```
What is the range to the target in meters?
>> 4500
Noting the-range = 4500 cf 100 because you said so.
Found the-range.
Seeking weapon-type(tank,extreme).
Using kb-191:
    weapon-type(tank,extreme) = indirect.
Noting weapon-type(tank,extreme) = indirect cf 100 because kb-191.
Found weapon-type(tank,extreme).
Noting weapon-selected = indirect cf 100 because kb-158.
kb-158 succeeded.
Found weapon-selected.
Seeking ammunition-selected.
Invoking kb-211:
    if weapon-selected = guided_missile
    then ammunition-selected = anti_tank.
Already sought weapon-selected.
kb-211 failed.
Invoking kb-212:
    if weapon-selected = automatic_cannon and
       the-target = uncertain and
       enemy-unit-type = R and
       ammunition-type(automatic_cannon,R) = Ammunition_return
    then ammunition-selected = Ammunition_return.
Already sought weapon-selected.
kb-212 failed.
Invoking kb-213:
    if weapon-selected = automatic_cannon and
       the-target = R and
       ammunition-type(automatic_cannon,R) = Ammunition_return
    then ammunition-selected = Ammunition_return.
Already sought weapon-selected.
kb-213 failed.
Invoking kb-214:
    if weapon-selected = machinegun
    then ammunition-selected = ball/tracer.
Already sought weapon-selected.
kb-214 failed.
```

```
Invoking kb-215:
    if weapon-selected = indirect and
       the-terrain is sought and
       the-terrain = T and
       the-target = uncertain and
       enemy-unit-type = R and
       artillery-type(T,R) = Artillery_return
    then ammunition-selected = Artillery_return.
Already sought weapon-selected.
Seeking the-terrain.
Using kb-187:
    question(the-terrain) =
    '\n \t What type of terrain is the target in? \n '.

      What type of terrain is the target in?

    1.  open
    2.  wooded
    3.  trenches/dug in
    4.  town
>> 1
Noting the-terrain = open cf 100 because you said so.
Found the-terrain.
Already sought the-terrain.
Already sought the-target.
kb-215 failed.
Invoking kb-216:
    if weapon-selected = indirect and
       the-terrain is sought and
       the-terrain = T and
       the-target = R and
       artillery-type(T,R) = Artillery_return
    then ammunition-selected = Artillery_return.
Already sought weapon-selected.
Already sought the-terrain.
Already sought the-terrain.
Already sought the-target.
Seeking artillery-type(open,tank).
Using kb-231:
    artillery-type(open,tank) = dual_purpose_improved_conventional_munition.
Noting artillery-type(open,tank) = dual_purpose_improved_conventional_munition cf 100 because
kb-231.
Found artillery-type(open,tank).
Noting ammunition-selected = dual_purpose_improved_conventional_munition cf 100 because kb-
216.
kb-216 succeeded.
Found ammunition-selected.
Seeking output-range.
```

```
Invoking kb-52:
    if mostlikely(the-range) = Range and
       display('\f \n \n \n \t The range to the target is ') and
       display(Range) and
       display(' meters.\n \n \n')
    then output-range.
Already sought the-range.
```

The range to the target is 4500 meters.

```
Noting output-range = yes cf 100 because kb-52.
kb-52 succeeded.
Found output-range.
Seeking output-weapon.
Invoking kb-53:
    if mostlikely(weapon-selected) = Weapon and
       Weapon == indirect and
       mostlikely(ammunition-selected) = Ammunition and
       display('\t The preferred weapon is artillery fire because \n') and
       display('\t of the long distance to the target.\n  \n') and
       display('\t Request indirect fire through your fire support channels.')
        and
       display('\n \t The preferred munition is ') and
       display(Ammunition)
    then output-weapon.
Already sought weapon-selected.
Already sought ammunition-selected.
```

The preferred weapon is artillery fire because
of the long distance to the target.

Request indirect fire through your fire support channels.
The preferred munition is dual purpose improved conventional munition

```
Noting output-weapon = yes cf 100 because kb-53.
kb-53 succeeded.
Found output-weapon.
Seeking the-user-is-finished.
Using kb-60:
    question(the-user-is-finished) =
    ['\n \t This concludes one iteration of the Bradley Gunner.  Do\n ',
    '\t you wish to continue?\n '].
```

```
      This concludes one iteration of the Bradley Gunner.  Do
      you wish to continue?

    1.  yes
    2.  no
>> 1

Noting the-user-is-finished = yes cf 100 because you said so.
Found the-user-is-finished.
Noting continue the-consultation = yes cf 100 because kb-44.
kb-44 succeeded.
Found continue the-consultation.
Seeking all-targets-engaged.
Invoking kb-46:
    if continue the-consultation = no and
       do(abort)
    then all-targets-engaged.
Already sought continue the-consultation.
kb-46 failed.
Invoking kb-47:
    if continue the-consultation and
       do(reset continue the-consultation) and
       do(reset self-defense) and
       do(reset should-fire) and
       do(reset the-target) and
       do(reset the-side) and
       do(reset the-terrain) and
       do(reset the-range) and
       do(reset weapon-selected) and
       do(reset ammunition-selected) and
       do(reset maybe-should-fire) and
       do(reset dismounted-check) and
       do(reset location-check) and
       do(reset formation-check) and
       do(reset unknown-near-road) and
       do(reset unknown-location) and
       do(reset unknown-multiple-target) and
       do(reset output-range) and
       do(reset output-weapon) and
       do(reset unknown-in-formation) and
       do(reset the-user-is-finished) and
       do(restart)
    then all-targets-engaged.
Already sought continue the-consultation.

Already sought begin the-consultation.
Seeking continue the-consultation.
Invoking kb-42:
    if should-fire = no and
       display('\f \n \n \n \t Do not shoot at friendly troops.\n \n ') and
```

```
       the-user-is-finished = Answer
    then continue the-consultation = Answer.
Seeking should-fire.
Invoking kb-43:
    if self-defense = yes and
       display('\n \t Firing in self defense is always authorized. \n')
    then should-fire = yes cf 100.
Seeking self-defense.
Using kb-56:
    question(self-defense) =
    ['\f \n \n \t \t Is the target currently firing at you? \n \n'].


          Is the target currently firing at you?


    1.  yes
    2.  no
>> 2
Noting self-defense = no cf 100 because you said so.
Found self-defense.
kb-43 failed.
Invoking kb-64:
    if the-side = enemy
    then should-fire = yes.
Seeking the-side.
Using kb-154:
    question(the-side) =
    '\n \tIs the target enemy, friendly, or not known?\n'.

     Is the target enemy, friendly, or not known?

    1.  enemy
    2.  friendly
    3.  not known
>> 2
Noting the-side = friendly cf 100 because you said so.
Found the-side.
kb-64 failed.
Invoking kb-65:
    if the-side is sought and
       the-side = friendly
    then should-fire = no.
Already sought the-side.
Already sought the-side.
Noting should-fire = no cf 100 because kb-65.
kb-65 succeeded.
Found should-fire.
```

```
     Do not shoot at friendly troops.

 Seeking the-user-is-finished.
Using kb-60:
    question(the-user-is-finished) =
    ['\n \t This concludes one iteration of the Bradley Gunner.  Do\n ',
    '\t you wish to continue?\n '].

     This concludes one iteration of the Bradley Gunner.  Do
     you wish to continue?

    1.  yes
    2.  no
>> 1
Noting the-user-is-finished = yes cf 100 because you said so.
Found the-user-is-finished.
Noting continue the-consultation = yes cf 100 because kb-42.
kb-42 succeeded.
Found continue the-consultation.
Seeking all-targets-engaged.
Invoking kb-46:
    if continue the-consultation = no and
       do(abort)
    then all-targets-engaged.
Already sought continue the-consultation.
kb-46 failed.
```

```
Invoking kb-47:
    if continue the-consultation and
       do(reset continue the-consultation) and
       do(reset self-defense) and
       do(reset should-fire) and
       do(reset the-target) and
       do(reset the-side) and
       do(reset the-terrain) and
       do(reset the-range) and
       do(reset weapon-selected) and
       do(reset ammunition-selected) and
       do(reset maybe-should-fire) and
       do(reset dismounted-check) and
       do(reset location-check) and
       do(reset formation-check) and
       do(reset unknown-near-road) and
       do(reset unknown-location) and
       do(reset unknown-multiple-target) and
       do(reset output-range) and
       do(reset output-weapon) and
       do(reset unknown-in-formation) and
       do(reset the-user-is-finished) and
       do(restart)
    then all-targets-engaged.
Already sought continue the-consultation.

Already sought begin the-consultation.
Seeking continue the-consultation.
Invoking kb-42:
    if should-fire = no and
       display('\f \n \n \n \t Do not shoot at friendly troops.\n \n ') and
       the-user-is-finished = Answer
    then continue the-consultation = Answer.
Seeking should-fire.
Invoking kb-43:
    if self-defense = yes and
       display('\n \t Firing in self defense is always authorized. \n')
    then should-fire = yes cf 100.
Seeking self-defense.
Using kb-56:
    question(self-defense) =
    ['\f \n \n \t \t Is the target currently firing at you? \n \n'].
```

```
          Is the target currently firing at you?

    1. yes
    2. no
>> 2
Noting self-defense = no cf 100 because you said so.
Found self-defense.
kb-43 failed.
Invoking kb-64:
    if the-side = enemy
    then should-fire = yes.
Seeking the-side.
Using kb-154:
    question(the-side) =
    '\n \tIs the target enemy, friendly, or not known?\n'.

     Is the target enemy, friendly, or not known?

    1. enemy
    2. friendly
    3. not known
>> 3
Noting the-side = not_known cf 100 because you said so.
Found the-side.
kb-64 failed.
Invoking kb-65:
    if the-side is sought and
       the-side = friendly
    then should-fire = no.
Already sought the-side.
Already sought the-side.
kb-65 failed.
Invoking kb-66:
    if the-side is sought and
       the-side = not_known and
       status-check = SC and
       do(set maybe-should-fire = yes cf SC) and
       dismounted-check = DC and
       do(set maybe-should-fire = yes cf DC) and
       location-check = LC and
       do(set maybe-should-fire = yes cf LC) and
       formation-check = FC and
       do(set maybe-should-fire = yes cf FC) and
       maybe-should-fire = Final
    then should-fire = Final.
Already sought the-side.
Already sought the-side.
Seeking status-check.
Invoking kb-67:
```

```
    if friendly-status = FS and
       enemy-status = ES and
       status-check-function(FS,ES) = SCF
    then status-check = SCF.
Already sought friendly-status.
Already sought enemy-status.
Seeking status-check-function(defending,attacking).
Using kb-70:
    status-check-function(defending,attacking) = 60.
Noting status-check-function(defending,attacking) = 60 cf 100 because kb-70.
Found status-check-function(defending,attacking).
Noting status-check = 60 cf 100 because kb-67.
kb-67 succeeded.
Found status-check.
Noting maybe-should-fire = yes cf 60 because kb-66.
Seeking dismounted-check.
Invoking kb-96:
    if the-target = UTT and
       patrol-status = PS and
       civilian-status = CS and
       dismounted-check-function(UTT,PS,CS) = DCF
    then dismounted-check = DCF.
Seeking the-target.
Using kb-114:
    question(the-target) = '\f \n \t What type of target is it? \n '.

      What type of target is it?

    1.  tank
    2.  apc
    3.  sp artillery
    4.  towed artillery
    5.  anti tank unit
    6.  infantry
    7.  trucks
    8.  logistics area
    9.  command post
    10.  tracked
    11.  wheeled
    12.  dismounted
    13.  uncertain
>> 1
Noting the-target = tank cf 100 because you said so.
Found the-target.
Already sought patrol-status.
Already sought civilian-status.
Seeking dismounted-check-function(tank,no,no).
Using kb-109:
    dismounted-check-function(tank,no,no) = 40.
Noting dismounted-check-function(tank,no,no) = 40 cf 100 because kb-109.
```

```
Found dismounted-check-function(tank,no,no).
Noting dismounted-check = 40 cf 100 because kb-96.
kb-96 succeeded.
Found dismounted-check.
Noting maybe-should-fire = yes cf 40 because kb-66.
Seeking location-check.
Invoking kb-118:
    if unknown-near-road = UNR and
       unknown-location = UL and
       location-check-function(UNR,UL) = LCF
    then location-check = LCF.
Seeking unknown-near-road.
Using kb-120:
    question(unknown-near-road) =
    '\f \n \t Is the unknown target on, near or away from a road?\n '.

      Is the unknown target on, near or away from a road?

    1. on
    2. near
    3. away
>> 3
Noting unknown-near-road = away cf 100 because you said so.
Found unknown-near-road.
Seeking unknown-location.
Using kb-125:
    question(unknown-location) =
    '\n \t Is the target near a known enemy or friendly location? \n '.

      Is the target near a known enemy or friendly location?

    1. enemy
    2. friendly
    3. neither
>> 1
Noting unknown-location = enemy cf 100 because you said so.
Found unknown-location.
Seeking location-check-function(away,enemy).
Using kb-131:
    location-check-function(away,enemy) = 30.
Noting location-check-function(away,enemy) = 30 cf 100 because kb-131.
Found location-check-function(away,enemy).
Noting location-check = 30 cf 100 because kb-118.
kb-118 succeeded.
Found location-check.
Noting maybe-should-fire = yes cf 30 because kb-66.
Seeking formation-check.
Invoking kb-138:
    if unknown-multiple-target = UMT and
       unknown-in-formation = UF and
```

```
       formation-check-function(UMT,UF) = FCF
    then formation-check = FCF.
Seeking unknown-multiple-target.
Using kb-140:
    question(unknown-multiple-target) =
    '\f \n Are there more than one unknown targets? \n '.

 Are there more than one unknown targets?

    1. yes
    2. no
>> 1
Noting unknown-multiple-target = yes cf 100 because you said so.
Found unknown-multiple-target.
Seeking unknown-in-formation.
Using kb-145:
    question(unknown-in-formation) = '\nIs the target(s) in a formation?\n '.

Is the target(s) in a formation?

    1. yes
    2. no
>> 1
Noting unknown-in-formation = yes cf 100 because you said so.
Found unknown-in-formation.
Seeking formation-check-function(yes,yes).
Using kb-152:
    formation-check-function(yes,yes) = 15.
Noting formation-check-function(yes,yes) = 15 cf 100 because kb-152.
Found formation-check-function(yes,yes).
Noting formation-check = 15 cf 100 because kb-138.
kb-138 succeeded.
Found formation-check.
Noting maybe-should-fire = yes cf 15 because kb-66.
Already sought maybe-should-fire.
Noting should-fire = yes cf 85 because kb-66.
kb-66 succeeded.
Found should-fire.
kb-42 failed.
Invoking kb-44:
    if self-defense is sought and
       should-fire is sought and
       should-fire cf N and
       N>= 80 and
       weapon-selected is sought and
       ammunition-selected is sought and
       output-range and
       output-weapon and
       the-user-is-finished = Answer
    then continue the-consultation = Answer.
```

```
Already sought self-defense.
Already sought should-fire.
Already sought should-fire.
Seeking weapon-selected.
Invoking kb-158:
    if (the-target = tank or
        (the-target = uncertain and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = infantry)) and
       the-range = Range and
       Range>= 3751 and
       weapon-type(tank,extreme) = Weapon_return
    then weapon-selected = Weapon_return.
Already sought the-target.
Seeking the-range.
Using kb-184:
    question(the-range) = '\n \t What is the range to the target in meters?'.

      What is the range to the target in meters?
>> 1250
Noting the-range = 1250 cf 100 because you said so.
Found the-range.
Already sought the-target.
Already sought the-target.
Already sought the-target.
kb-158 failed.
```

```
Invoking kb-159:
    if (the-target = tank or
        (the-target = uncertain and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = tank) or
        (the-target = tracked and
         enemy-unit-type = infantry)) and
       the-range = Range and
       Range<3751 and
       Range>= 65 and
       weapon-type(tank,long) = Weapon_return
    then weapon-selected = Weapon_return.
Already sought the-target.
Already sought the-range.
Seeking weapon-type(tank,long).
Using kb-192:
    weapon-type(tank,long) = guided_missile.
Noting weapon-type(tank,long) = guided_missile cf 100 because kb-192.
Found weapon-type(tank,long).
Noting weapon-selected = guided_missile cf 100 because kb-159.
kb-159 succeeded.
Found weapon-selected.
Seeking ammunition-selected.
Invoking kb-211:
    if weapon-selected = guided_missile
    then ammunition-selected = anti_tank.
Already sought weapon-selected.
Noting ammunition-selected = anti_tank cf 100 because kb-211.
kb-211 succeeded.
Found ammunition-selected.
Seeking output-range.
Invoking kb-52:
    if mostlikely(the-range) = Range and
       display('\f \n \n \n \t The range to the target is ') and
       display(Range) and
       display(' meters.\n \n \n')
    then output-range.
Already sought the-range.


     The range to the target is 1250 meters.


Noting output-range = yes cf 100 because kb-52.
kb-52 succeeded.
Found output-range.
Seeking output-weapon.
Invoking kb-53:
```

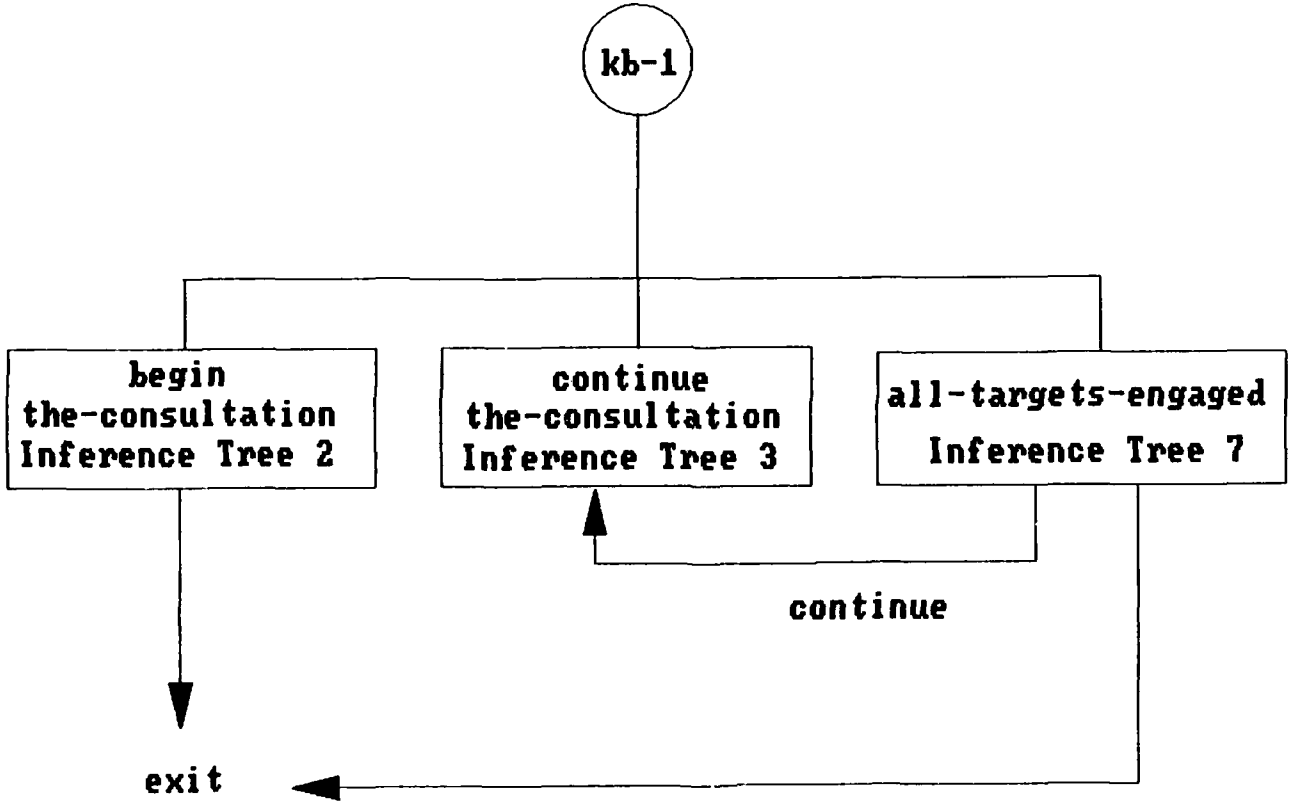
```
    if mostlikely(weapon-selected) = Weapon and
       Weapon == indirect and
       mostlikely(ammunition-selected) = Ammunition and
       display('\t The preferred weapon is artillery fire because \n') and
       display('\t of the long distance to the target.\n  \n') and
       display('\t Request indirect fire through your fire support channels.')
        and
       display('\n \t The preferred munition is ') and
       display(Ammunition)
    then output-weapon.
Already sought weapon-selected.
kb-53 failed.
Invoking kb-54:
    if mostlikely(weapon-selected) = Weapon and
       mostlikely(ammunition-selected) = Ammunition and
       display('\t The preferred weapon is the ') and
       display(Weapon) and
       display('\n \t firing ') and
       display(Ammunition) and
       display(' ammunition.')
    then output-weapon.
Already sought weapon-selected.
Already sought ammunition-selected.
      The preferred weapon is the guided_missile
      firing anti_tank ammunition.Noting output-weapon = yes cf 100 because kb-54.
kb-54 succeeded.
Found output-weapon.
Seeking the-user-is-finished.
Using kb-60:
    question(the-user-is-finished) =
    ['\n \t This concludes one iteration of the Bradley Gunner.  Do\n ',
    '\t you wish to continue?\n '].

      This concludes one iteration of the Bradley Gunner.  Do
      you wish to continue?

    1.  yes
    2.  no
>> 2
Noting the-user-is-finished = no cf 100 because you said so.
Found the-user-is-finished.
Noting continue the-consultation = no cf 100 because kb-44.
kb-44 succeeded.
Found continue the-consultation.
Seeking all-targets-engaged.
Invoking kb-46:
    if continue the-consultation = no and
       do(abort)
    then all-targets-engaged.
Already sought continue the-consultation.
```

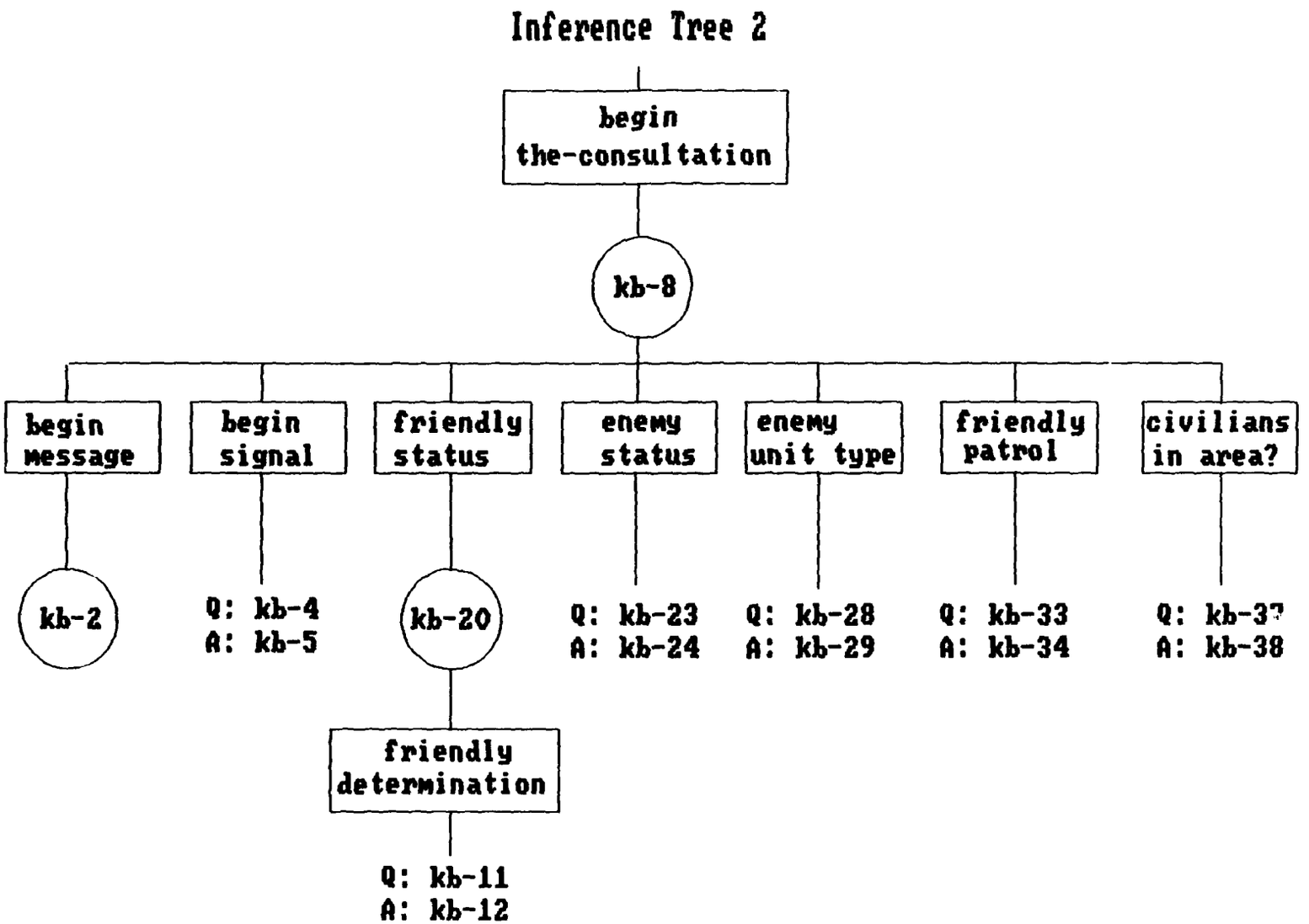
```
M.1> log off
```

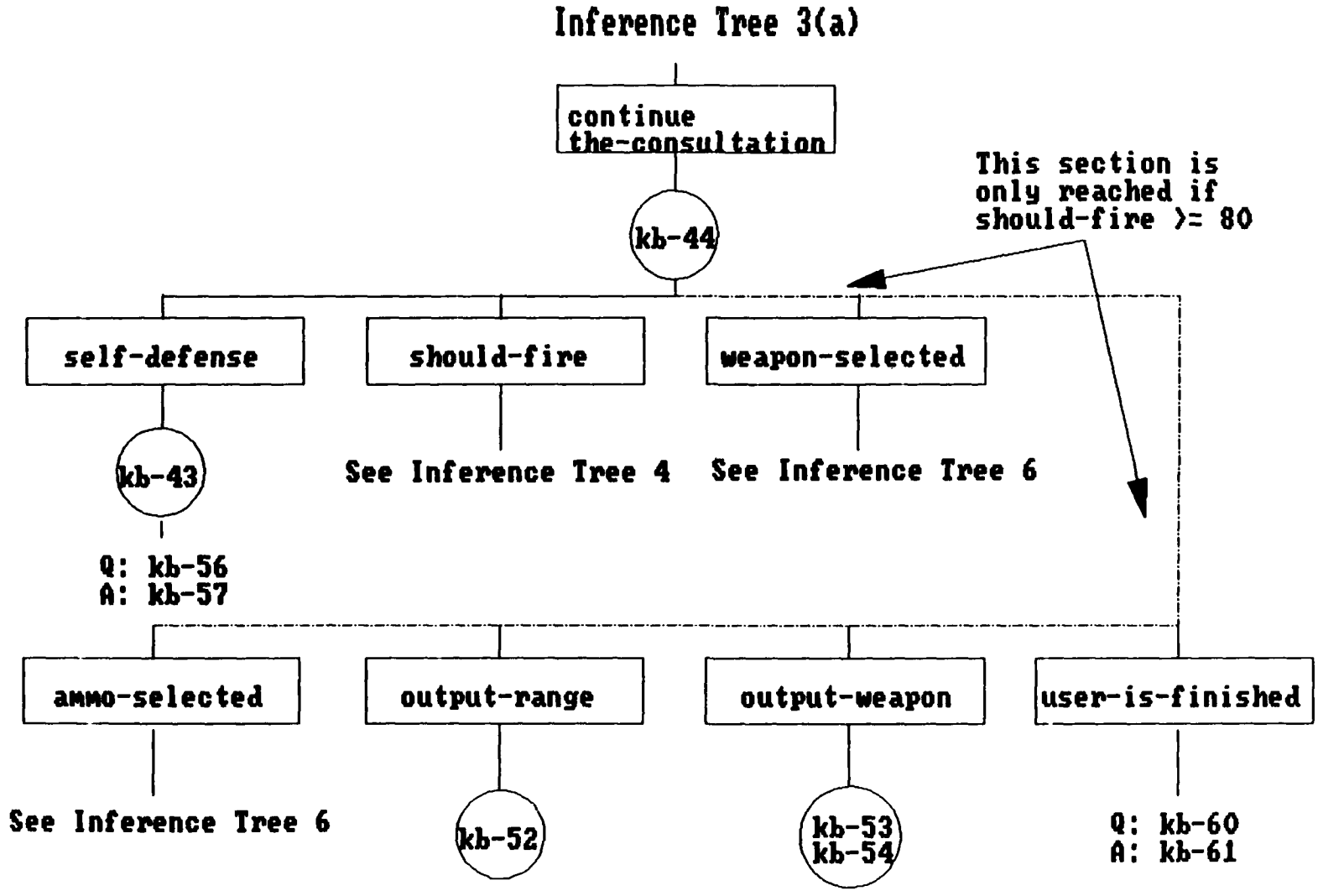
APPENDIX V

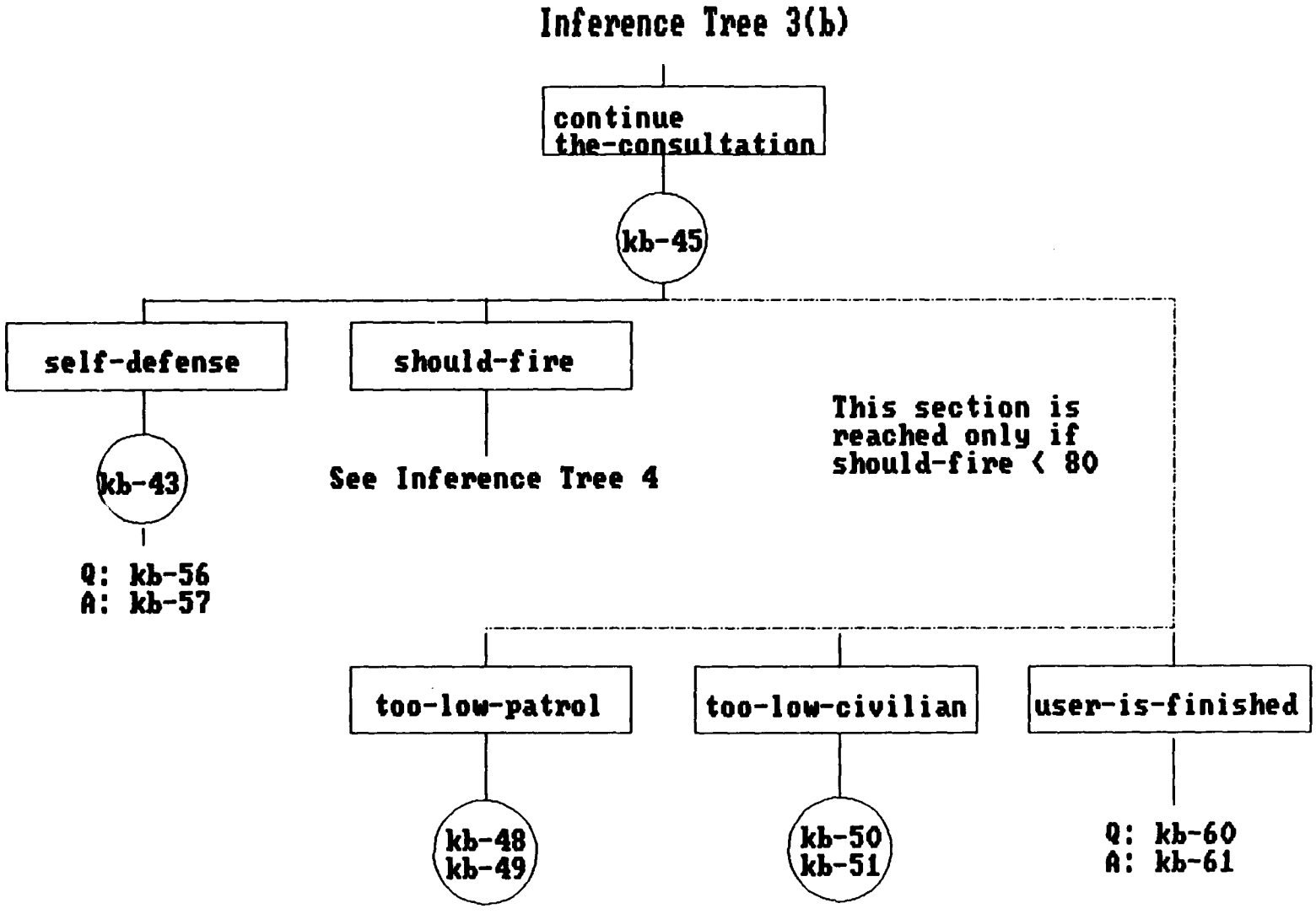
Appendix V is a representation of the expert gunnery system as an inference tree. The rules referred to here as "kb-xx" can be found in their entirety in Appendix II.

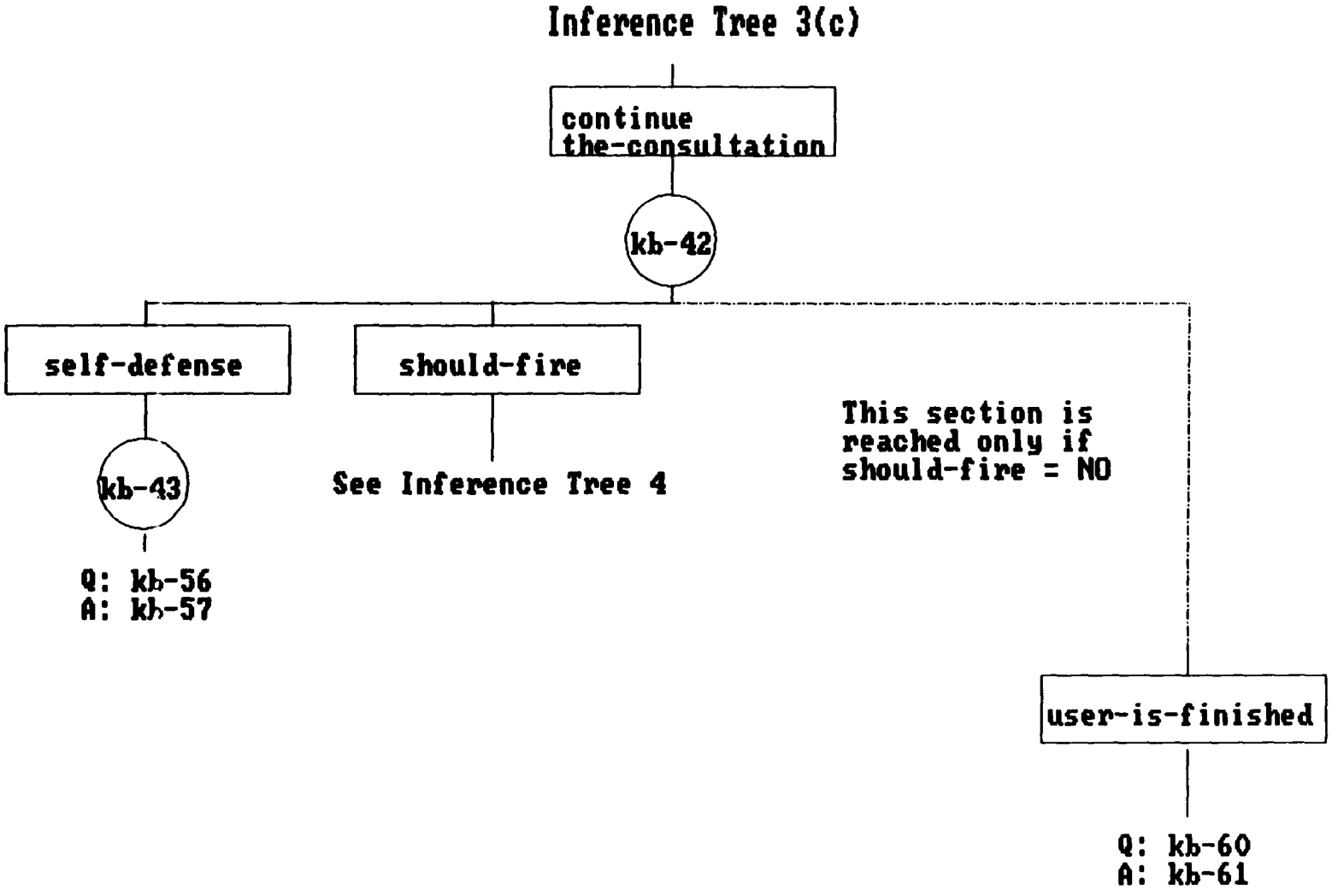
Inference Tree 1



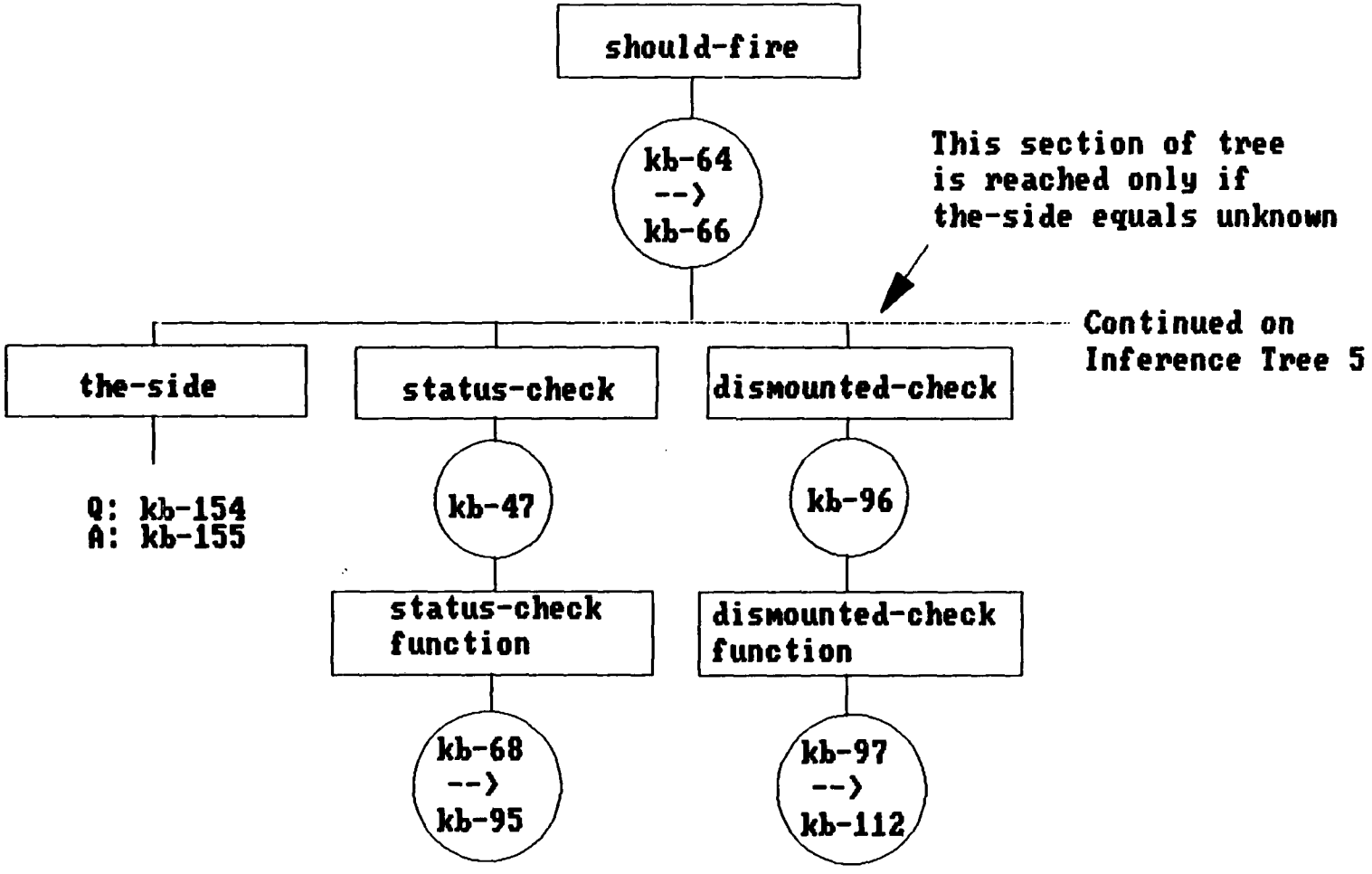

Inference Tree 2
begin the-consultation
kb-8
begin message
begin signal
friendly status
enemy status
enemy unit type
friendly patrol
civilians in area?
kb-2
Q: kb-4
A: kb-5
kb-20
Q: kb-23
A: kb-24
Q: kb-28
A: kb-29
Q: kb-33
A: kb-34
Q: kb-37
A: kb-38
friendly determination
Q: kb-11
A: kb-12


Inference Tree 3(a)
continue the-consultation
kb-44
This section is only reached if should-fire >= 80
self-defense
should-fire
weapon-selected
kb-43
See Inference Tree 4
See Inference Tree 6
Q: kb-56
A: kb-57
ammo-selected
output-range
output-weapon
user-is-finished
See Inference Tree 6
kb-52
kb-53
kb-54
Q: kb-60
A: kb-61


Inference Tree 3(b)
continue the-consultation
kb-45
self-defense
should-fire
This section is reached only if should-fire < 80
kb-43
See Inference Tree 4
Q: kb-56
A: kb-57
too-low-patrol
too-low-civilian
user-is-finished
kb-48
kb-49
kb-50
kb-51
Q: kb-60
A: kb-61

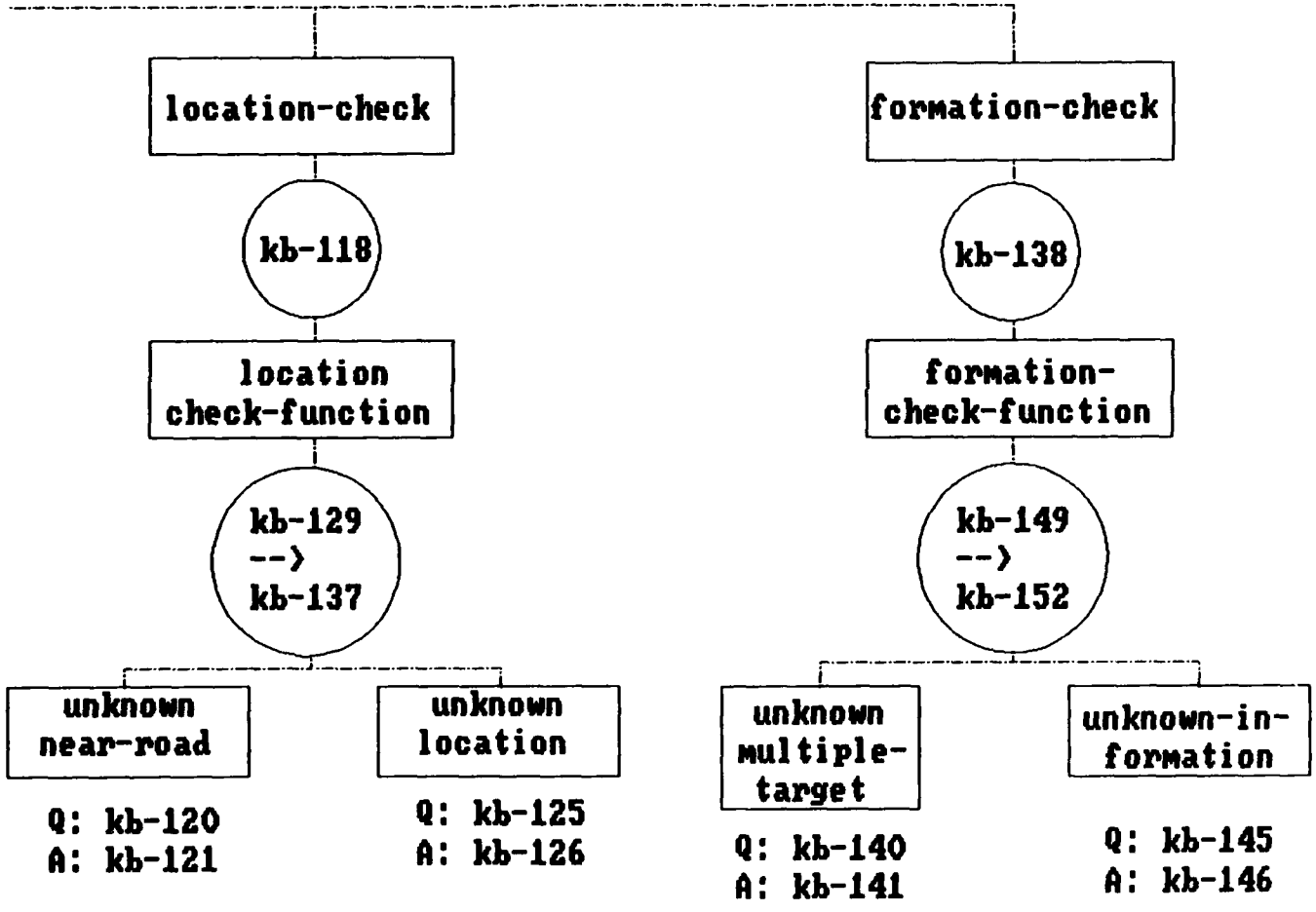

Inference Tree 3(c)
continue the-consultation
kb-42
self-defense
should-fire
This section is reached only if should-fire = NO
kb-43
See Inference Tree 4
Q: kb-56
A: kb-57
user-is-finished
Q: kb-60
A: kb-61

Inference Tree 4



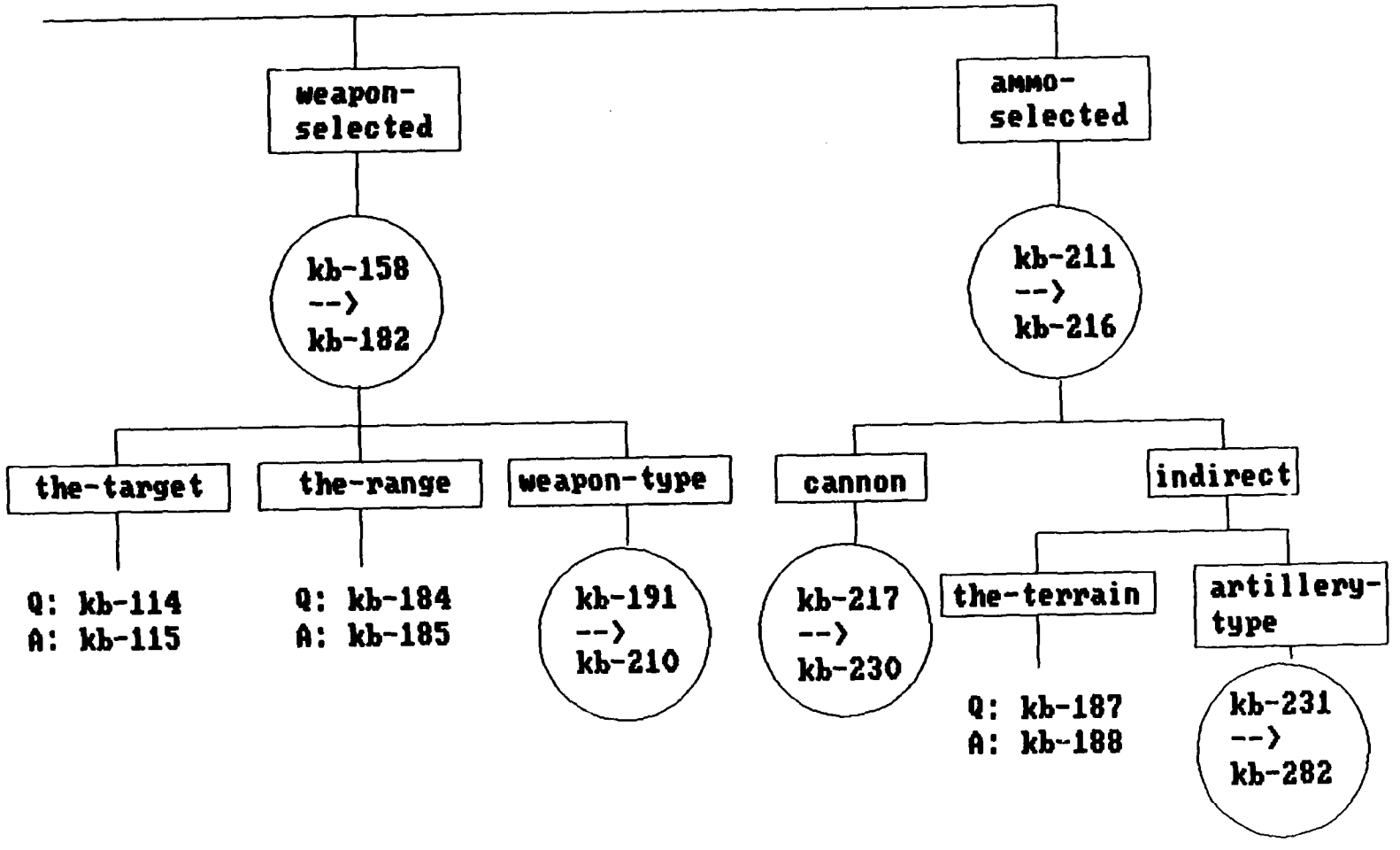
Inference Tree 5

Continued from
Inference Tree 4

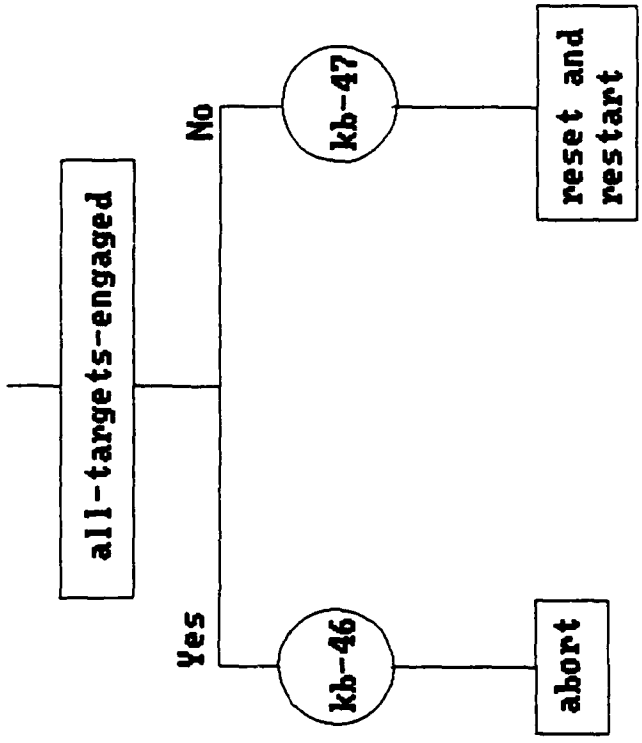


Inference Tree 6

Continued from
Inference Tree 3(a)



Inference Tree 7



END

FILMED

7-89

DTIC